CS CAPSTONE PROJECT ARCHIVE

MAY 29, 2020

# AI THREAT INTELLIGENCE


PREPARED BY

GROUP 4

TEAM ART

NIKHIL ANAND

ROHAN VARMA

HARSHVARDHAN SINGH



**Abstract**

In this document we will discuss the progress that our group made over the last three terms. We will include the status of the various components that make up our project and discuss what work still needs to be done to complete the project.

# CONTENTS


**1 Introduction** 4

1.1 The Team . . . 4

1.2 Spring Term Changes . . . 4

**2 Requirements Document** 4

2.1 . . . 4

2.2 Change Table . . . 11

**3 Design Document** 11

3.1 . . . 11

3.2 Change Table . . . 22

**4 Weekly Blog posts for Fall and Winter** 22

4.1 Nikhil Anand . . . 22

4.1.1 October 17, 2019 . . . 22

4.1.2 October 25, 2019 . . . 22

4.1.3 November 1, 2019 . . . 22

4.1.4 November 8, 2019 . . . 22

4.1.5 November 15, 2019 . . . 23

4.1.6 November 22, 2019 . . . 23

4.1.7 November 29, 2019 . . . 23

4.1.8 January 10, 2020 . . . 23

4.1.9 January 17, 2020 . . . 24

4.1.10 January 24, 2020 . . . 24

4.1.11 February 7, 2020 . . . 24

4.1.12 February 14, 2020 . . . 24

4.1.13 February 21, 2020 . . . 25

4.1.14 February 28, 2020 . . . 25

4.1.15 March 6, 2020 . . . 25

4.1.16 March 13, 2020 . . . 25

4.2 Rohan Varma . . . 26

4.2.1 October 17, 2019 . . . 26

4.2.2 October 25, 2019 . . . 26

4.2.3 November 1, 2019 . . . 26

4.2.4 November 8, 2019 . . . 26

4.2.5 November 22, 2019 . . . 26

4.2.6 November 29, 2019 . . . 26

4.2.7 January 10, 2020 . . . 27

4.2.8 January 17, 2020 . . . 27

4.2.9 January 24, 2020 . . . 27
4.2.10 February 7, 2020 . . . 27
4.2.11 February 14, 2020 . . . 27
4.2.12 February 21, 2020 . . . 27
4.2.13 February 28, 2020 . . . 28
4.2.14 March 6, 2020 . . . 28
4.2.15 March 13, 2020 . . . 28
4.3 Harshvardhan Singh . . . 28
4.3.1 October 25, 2019 . . . 28
4.3.2 November 1, 2019 . . . 28
4.3.3 November 8, 2019 . . . 29
4.3.4 November 15, 2019 . . . 29
4.3.5 November 22, 2019 . . . 29
4.3.6 November 29, 2019 . . . 29
4.3.7 February 8,2020 . . . 29
4.3.8 February 14, 2020 . . . 30
4.3.9 February 21, 2020 . . . 30
4.3.10 February 28, 2020 . . . 30
4.3.11 March 4, 2020 . . . 30
4.3.12 March 13, 2020 . . . 30

**5 Tech Reviews** 31
5.1 Nikhil Anand . . . 31
5.2 Rohan Varma . . . 38
5.3 Harshvardhan Singh . . . 43

**6 Project Documentation** 49
6.1 How it works . . . 49
6.2 Installation . . . 51
6.3 Running the Pipeline . . . 53
6.4 Useful Links and API documentation . . . 53

**7 Recommended Technical Resources for Learning More** 53

**8 Poster** 53

**9 Conclusions and Reflections** 55
9.1 Nikhil Anand . . . 55
9.2 Rohan Varma . . . 55
9.3 Harshvardhan Singh . . . 55

**10 Appendix** 56

10.1 Appendix 1 . . . . . 56
10.2 Appendix 2 . . . . . 57
10.3 Appendix 3 . . . . . 58

# 1 Introduction

Attackers are constantly looking for ways to exploit the latest vulnerabilities and/or new ways to exploit old vulnerabilities. Cyber-attacks are often leveraged by threat actors as part of a coordinated campaign against a specific target. These campaigns usually have an object or goal with these campaigns being organized by threat actors from other nation-states, nefarious organizations or other crime syndicates. Attacks from these usually consist of similar properties, behaviors, and attributes for achieving their objectives over a noteworthy period of time. With over 400 new threats being discovered every minute, more and more innovative ways for disclosing threats have come up. One of the newer methods involves discussing such over social media sites such as Twitter and blogs. This project will require creating a system and method to identify the latest threat campaigns in the order of their relevance to enterprises. It will involve applying machine learning on the content of twitter to discover new threats and uncover potential threat campaigns. This project was brought to us by McAfee, LLC which is the worlds largest technology security company. This project was requested to optimize the research process that McAfee researches currently undergo. Currently for discovering new campaigns, researchers often browse news sites and twitter updates from known security researchers. Next they carryout link analysis and pivot on these discovered pieces of Open Source Intelligence and correlate it with other related artifacts that they may already have in their threat information repositories. Using this approach, they build and curate an enriched threat intelligence advisory for their customers. This project will create an automated pipeline based on machine learning to identify tweets reducing time for researching findings for new threat campaigns.

## 1.1 The Team

The clients for this project were Prashanth Ramagopal and Ezequias Simon. Both are senior engineers at McAfee with a background in mathematics and machine learning. The team members for this project were Rohan Varma, Harshvardhan Singh, and Nikhil Anand. Each team member worked on various aspects of the project, with Rohan taking a lead on the Machine Learning portion of the project, Harshvardhan worked on the natural language processing and graph database portion, and Nikhil worked on creating the front end tool used for dataset creation, and constructed the pipeline. Our clients involvement was instrumental to the success of this project, as we often had weekly meetings where we showed what progress had been made and discussed what challenges we were facing and were able to get valuable feedback to help overcome any hurdles.

## 1.2 Spring Term Changes

The changes for our course over the spring term didn't actually have an affect on our spring term deliverables, as we had modularized our duties for the project, we could all work on the project separately without being together. We also were all very comfortable with git which allowed us to all be on the same page and view each others progress. Our project is complete so we will not be needing another team of software engineers to continue our work.

# 2 Requirements Document

Next Page

## 2.1

# AI Threat Intelligence

Nikhil Anand, Harshvardhan Singh, Rohan Varma

Team ART (Automated Research Team)

***Abstract*— The client is McAfee, a cybersecurity company. As such, they have to perform constant research in order to keep up with the latest threats. The project aims to streamline their research process. The client has informed us that many cybersecurity researchers tweet about their findings. Accordingly, the project goal is to create a system that scans through tweets and points the clients towards new posts relevant to developments in cybersecurity research. The project will be deemed successful if the system is easily navigable and identifies relevant tweets with an accuracy of 80%.**

# Table of Contents

1. Introduction....3
   i. Premise....3
   ii. Scope....3
   iii. Definitions, Acronyms, and Abbreviations....3
2. Overall Description....3
   i. Product Features....3
   ii. Intended Users....3
   iii. Design and Implementation Constraints....4
3. System Requirements....4
   i. Scraping....4
   ii. Database....4
   iii. Data Curation....4
   iv. Tweet Classifier....5
   v. UI....5
4. Performance requirements....5
   i. Qualitative Requirements....5
   ii. Quantitative Requirements....5
5. Gantt Chart....6

# 1. Introduction

## i. *Premise*

The client for this project is McAfee, a company that focuses on cybersecurity. They produce products that protect computing systems against malware and other threats. One of the largest challenges they face is ensuring that their products are up to date and capable of defending against the latest threats. To meet these requirements in a rapidly evolving cybersecurity landscape, the client continuously performs thorough research; needless to say, this constant search for relevant news is an arduous task. Our aim for this project is to make this research process less labor intensive by automating portions of it.

## ii. *Scope*

Automating the client's research process is a very broad problem, and we will be focusing on a specific portion of it. The client has indicated to us that the cybersecurity community often uses Twitter as a platform to notify others about new developments in the field. However, there are millions of new tweets every day and only a small portion of them are relevant to the client. As such, we will be creating a tool that performs the first round of filtering for these tweets. Our goal will be to create an application that scans Twitter and gives the client a list of tweets that are related to new developments in cybersecurity research.

## iii. *Definitions, Acronyms, and Abbreviations*

IoC - Indicators of Compromise are pieces of forensic data that are used to identify malicious activity within a system or network. Some examples of these indicators may be unusual web traffic, increase in database read volume and suspicious file changes.

API - Application Programming Interface is the interface used for communication between a client and a server. Often used for client side software development.

UI - User Interface is the visual elements that users view and interact with. This includes all the buttons, icons, and pages that a user may see on a web interface.

AWS - Amazon Web Services is Amazon's on demand cloud platform.

GCP - Google Cloud Platform is a suite of cloud computing services offered by Google.

Pelican Cluster - Pelican Cluster is Oregon State Universities Linux based GPU cluster offered to students for remote computation for machine learning applications.

SOCMINT - Social Media Intelligence refers to the collective tools and techniques that companies use to monitor social media sites and extract useful data about emerging threats and opportunities to gain a competitive advantage.

TWINT - TWINT is an advanced Twitter scraping tool built in Python that allows users to scrape tweets from Twitter profiles without using the official Twitter API.

# 2. Overall Description

## i. *Product Features*

To meet the project goals, the system will have to have the following capabilities:

1. Gain access to tweets on Twitter and tabulate information such as content, poster, user, date, etc.
2. Store information on large amounts of tweets
3. Filter for tweets relevant to the client's cybersecurity focus
4. Display relevant tweets in an easily navigable UI

## ii. *Intended Users*

The intended users for our project are McAfee engineers who are utilizing our platform for searching Twitter for emerging threats that would be actionable on their part. These engineers are looking for information about threats that McAfee should be aware of, and would be able to improve their time to response against these threats

*iii. Design and Implementation Constraints*

A. Model Accuracy
   a) The accuracy of the model is directly related to the quality and size of the dataset.
   b) Manually creating the dataset and classifying tweets as relevant and irrelevant can introduce bias to the model as data curators are not experts in security.
   c) There will be noise in the dataset due to differences in opinions about tweet classification.

B. Compute Resources
   a) The training efficiency will depend directly on the compute power granted to it by McAfee or Oregon State University.
   b) Slow training time will lead to less modifications being tested which may prevent us from fully optimizing our model.

## 3. System Requirements

In order to meet the product features described above, the end system will have a scraper, data store, tweet classifier, and user interface. Requirements governing what is expected from each of these components are provided below. In order to train the tweet classifier, a training dataset must be acquired. As the data curation process leading to this is also governed by requirements, they have also been listed below.

*i. Scraping*

A. Summary

This component will be utilizing a scraping tool to bring tweets into the database. It will initially be used procure training data for the model, and later to find tweets for the model to classify

B. Functional Requirements
   a) Scraper can filter tweets based off of hashtags or specified keywords
   b) Scraper will have a dictionary of keywords and hashtags relevant to security tweets
   c) Scraper can search for tweets within a specified time range
   d) Scraper can access tweet content and user information
   e) Scraper can send search results to a specified database

*ii. Database*

A. Summary

This component will be the repository for tweets the scraper finds as well as a repository for tweets the model classifies as relevant. There will be two databases, one that holds training data for the model, and one for the end product that holds tweets that are to be classified by the model.

B. Functional Requirements
   a) Each database can store at least 30000 tweets
   b) Database can distinguish between unclassified and classified tweets
   c) Database can communicate with scraper, model, and UI

*iii. Data Curation*

A. Summary

Although not a system component, data curation is a critical process as it is how we will acquire our training set for the model. To ensure efficiency, initial data will be procured by having the scraper filter based off of specified keywords and hashtags. Then, the team will manually label these tweets as relevant or irrelevant

B. Process Requirements
   a) Team members labelling data must be trained to understand what McAfee considers relevant in the security space
   b) Keywords and hashtags used to provide initial filtering of tweets should be informed by McAfee's current research process
   c) Amount of training data should be commensurate with performance requirements for model accuracy. A minimum of 10000 tweets should be procured for training
   d) Training data should include both relevant and irrelevant data

*iv. Tweet Classifier*

A. Summary

This component will handle classifying the tweets as relevant or irrelevant. It will be trained utilizing the dataset obtained via the data curation process outlined above

B. Functional Requirements

a) Model can classify tweets as relevant or irrelevant with 80% accuracy on test sets
b) Model can draw inputs from database
c) Model can send classification results back to appropriate tables in database

*v. Noun Extractor*

A. Summary

This component will fetch singular proper nouns from tweets that have been classified as relevant.

B. Functional Requirements

a) Singular proper nouns from tweets can be retrieved
b) Nouns retrieved are all lower case without special characters
c) Function can send this information to graph database

*vi. Graph Database*

A. Summary

This component will store all the relevant tweets along with their relations to other tweets and replace the SQL database for production

B. Functional Requirements

a) Tweets have relevant relations based on keywords
b) Tweets have relations with author nodes for time tweeted
c) Database is not bulky with generic relations
d) Neo4j visualizer will be UI for developmental purposes

## 4. PERFORMANCE REQUIREMENTS

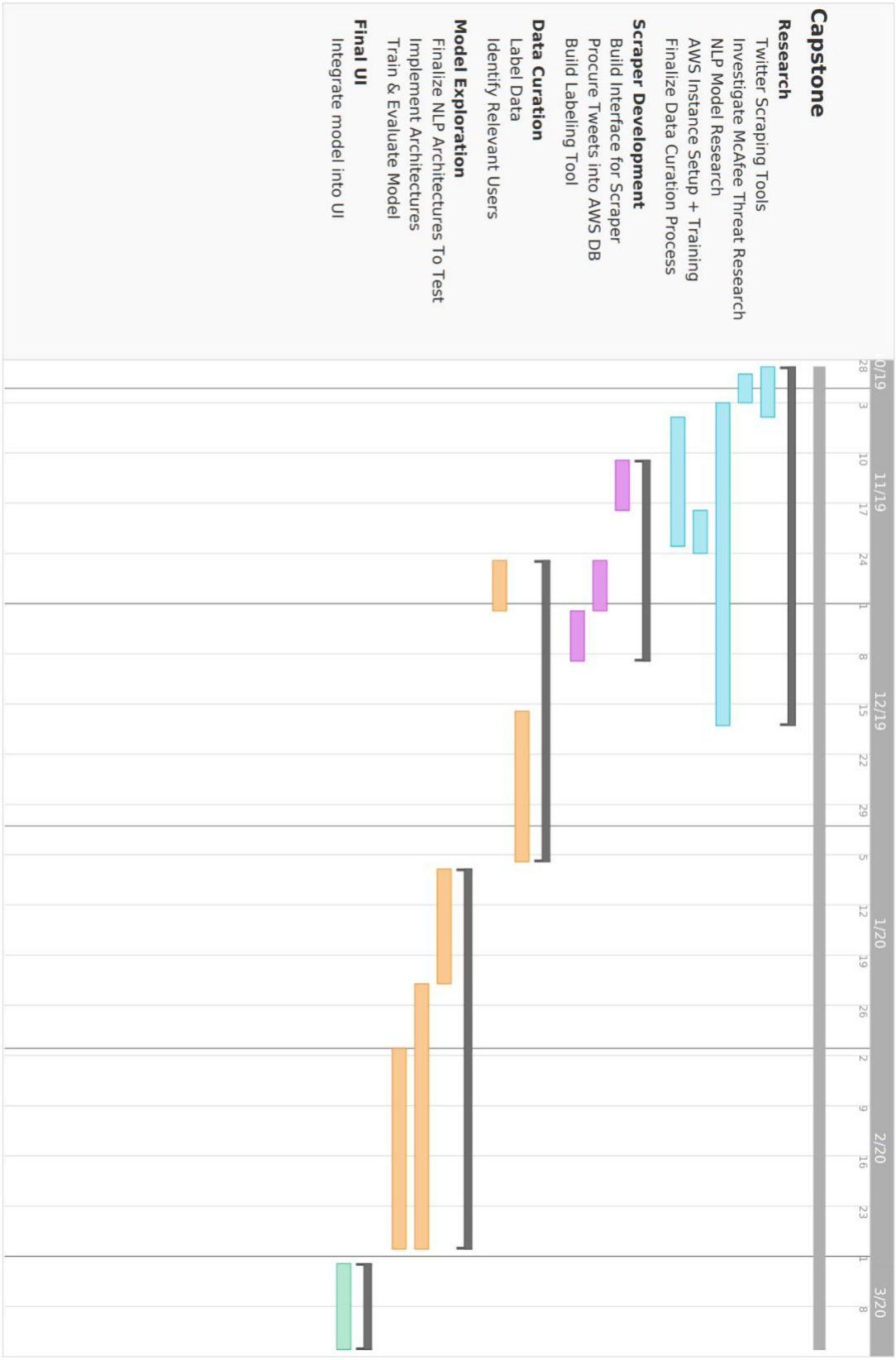
*i. Qualitative Requirements*

The main qualitative metric will be the "quality" of the feed provided to the user after classification. If the system can find tweets that a threat researcher thinks is actionable intelligence for research then the system is meeting expectations.

*ii. Quantitative Requirements*

A. The system should classify tweets relevance with at least 80% accuracy on test sets
B. The system shouldn't take more than 30 seconds to search, classify, and display a feed for tweets in a week's time range.

## 5. Gantt Chart



### 2.2 Change Table

1. Removed UI from System Components and Requirements 2. Added Noun Extractor and Graph Database

## 3 Design Document

### 3.1

Next Page

# CS CAPSTONE DESIGN DOCUMENT

NOVEMBER 24, 2019

# AI THREAT INTELLIGENCE

PREPARED FOR

## MCAFEE LLC

PREPARED BY

## GROUP 4

## AUTOMATED RESEARCH TEAM

NIKHIL ANAND
HARSHVARDHAN SINGH
ROHAN VARMA

**Abstract**

The problem that we are trying to solve is that currently researchers and engineers at McAfee are using precious work hours to scour the web for new and relevant threat campaigns, malware strains, and other cyber security related events. Once they have identified which threats are relevant and worth prioritizing they begin to understand the threats and reverse engineer them to make sure that McAfee is able to identify and protect against these threats in the best ways possible. This design document will discuss the various design decisions and viewpoints relevant to this project, and how we will approach the implementation of various design entities.

## CONTENTS

**1 Introduction** 2

1.1 Scope . . . . . 2

1.2 Purpose . . . . . 2

**2 Design Description** 2

2.1 Design Stakeholders . . . . . 2

2.2 Design Viewpoints . . . . . 2

2.2.1 Context Viewpoint . . . . . 2

2.2.2 Composition Viewpoint . . . . . 3

2.2.3 Algorithm Viewpoint . . . . . 5

**3 Approach** 5

3.1 Twitter Scraper and Data Curation . . . . . 5

3.2 Natural Language Processing Model . . . . . 5

3.3 Web Scraper and Clustering . . . . . 6

3.4 User Interface . . . . . .. . . 7

**References** 8

# 1 INTRODUCTION

## 1.1 Scope

This document includes the critical design choices that have been made for this capstone project, as well as various viewpoints that are relevant to the design, and finally the approaches we will use for various components.

## 1.2 Purpose

The problem that we are trying to solve is that currently researchers and engineers at McAfee are using precious work hours to scour the web for new and relevant threat campaigns, malware strains, and other cyber security related events. Once they have identified which threats are relevant and worth prioritizing they begin to understand the threats and reverse engineer them to make sure that McAfee is able to identify and protect against these threats in the best ways possible. Our job is to automate the initial part of this process, in order to provide McAfee engineers with relevant information and tweets without having to search through Twitter and other social media websites.

# 2 DESIGN DESCRIPTION

## 2.1 Design Stakeholders

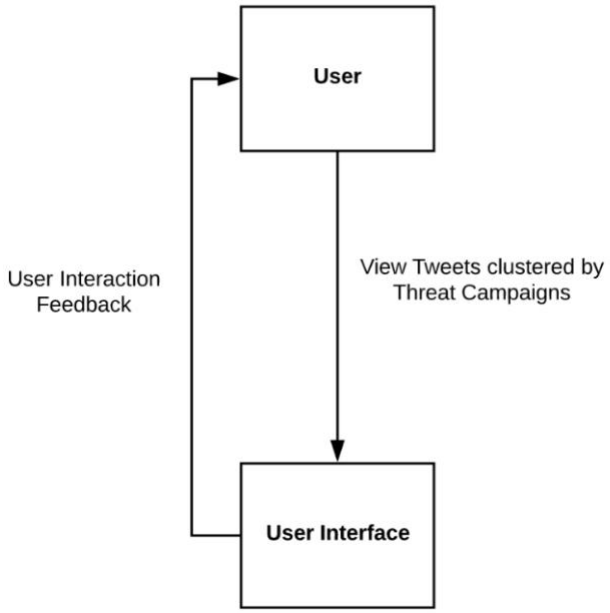
Prashanth Ramagopal and Sameer Paranpe: Prashanth is a Principal Engineer and Sameer is a Senior Security Researcher in the Enterprise group at McAfee. Both Sameer and Prashanth work on creating new threat detection methods and systems. They created this project to have an automated method for finding new threat campaigns using social media as often researchers post their findings about new security incidents on Twitter that lead to further global research/discovery of these incidents.

## 2.2 Design Viewpoints

### 2.2.1 Context Viewpoint

The context viewpoint for this project pertains to the interactions between users, who are security analysts and the final user interface that we create. Behind the scenes we will have a tweet scraper, a deep learning model, and a web scraper that will all work together to store relevant information in our database which will be used to display information on our user interface. But all these functionalities should be hidden to our users as is shown in the context viewpoint diagram below. The main priority of this context viewpoint is to emphasise that the user interface is efficient and simple to use. This means that it will use a minimalistic design, and perform user acceptance testing to ensure that the front end is meeting our criteria. This context is important because it shows that while the complete design of our project has many complex parts, the user should be shielded from those components to enhance their experience.

Fig. 1. Context Viewpoint Diagram




### 2.2.2 Composition Viewpoint

The main design entities of this project are: 1. Tweet Scraper

2. Deep Learning Model
3. Web Scraper
4. Database
5. Graph Database

This viewpoint is critical to explain the main components of the project and how each of them interact to produce the intended results. A streamlined design is important because it makes it easier for changes and enhancements to be made once the project has been completed. The success of the design will be based on how easily group members will be able to understand the different components of the project, and diagnose any problems that may arise on sections that are not under their direct control. The figures below will show both a high level and low level explanation for how our design entities come together.

Fig. 2. Low Level Component Diagram

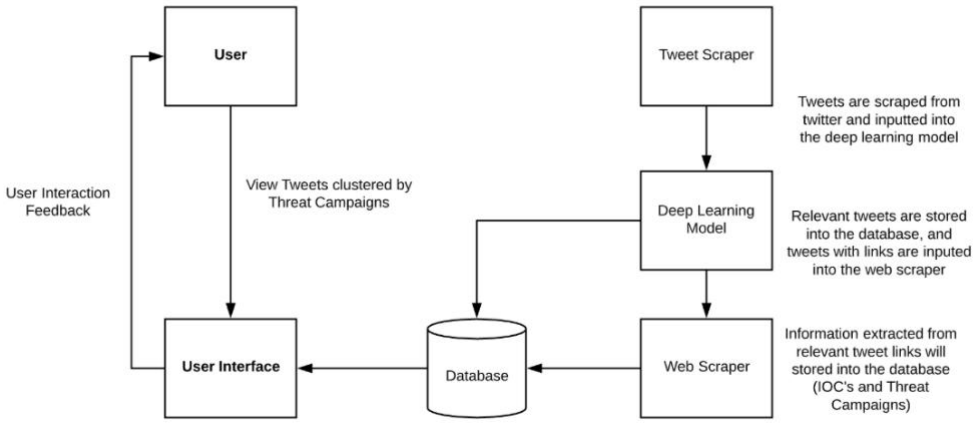
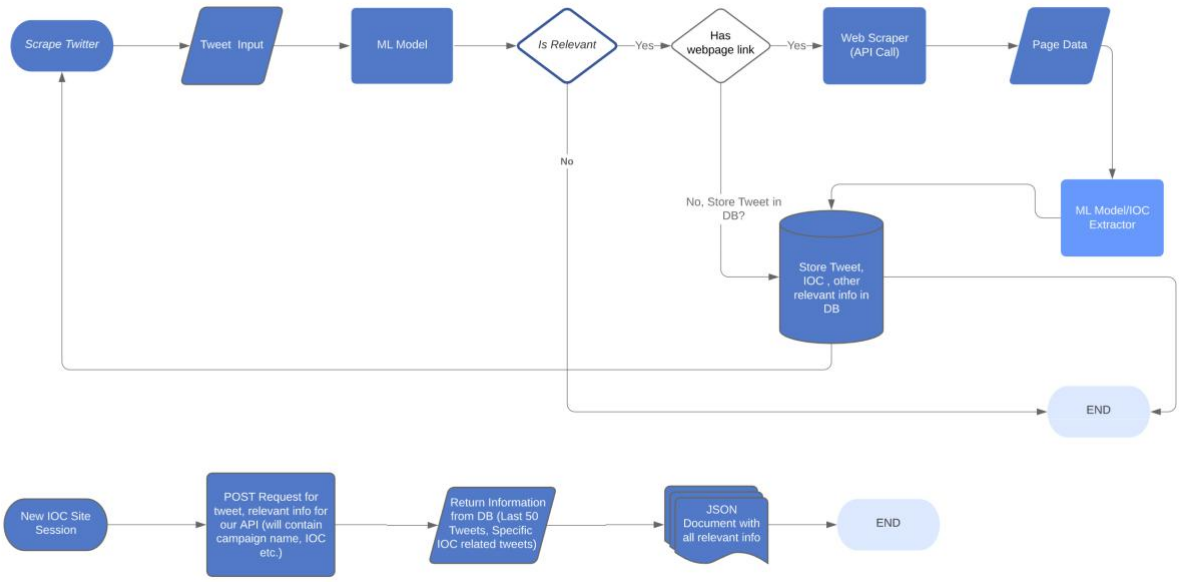



Fig. 3. High Level Component Diagram



#### 2.2.3 Algorithm Viewpoint

In this section we will be discussing our algorithm choice for the deep learning model and the clustering part of this project. The first task that we will be solving is to classify tweets as either relevant or irrelevant in regards to cybersecurity intelligence for McAfee. We will be modifying the BERT model, which is Google's Bidirectional Encoder Representations from Transformers. The algorithm as well as the rationale behind why we chose it will be further explained in our natural language processing model section. The analytical methods we will use to ensure that our model is working correctly will be to manually compare classifications from the BERT model with what a McAfee security analyst would actually classify the tweets as. The second task that we will be needing an algorithm for is for clustering the relevant tweets. For this part of the project we will be using NLTK and tf-idf. This works by finding nouns from a tweet and then only selecting relevant nouns from the previous list. For in depth details on how the algorithm works refer to Noun Extracting and Clustering. We will then store the noun, author as nodes in a graph database with number of occurrences of noun per day over time and relations between tweets and authors. The accuracy of our model and clustering is critical to the success of this project, which is why this viewpoint has been included in the document.

## 3 APPROACH

### 3.1 Twitter Scraper and Data Curation

Concern: Twint will be used to obtain Tweets for our data curation. The main challenge with this is that Twitter being a social media platform has a plethora of Tweets from different categories. Narrowing our search using different filters will be a must. This will require finding relevant attributes in cyber security related Tweets. After cyber security related Tweets have been obtained, they will be stored in our training table in our database. There will be another webpage that will be used to pull Tweets from the training table in our database for labeling them. Labeling Tweets as relevant and irrelevant is a manual process and will require us to know exactly what our client deems as relevant and irrelevant.

Approach: The Twint API will return batches of tweets with our specified filters which will be stored in the training database. Filters will be comprised of common jargons in security and also certain famous cyber-security researchers. Python will be used for running Twint and pulling Tweets, and also for connecting to the database using the same language. The webpage will be created using Javascript which will connect to the database and pull the new set of Tweets that require to be labelled. Once labelled, they will be stored in the database.

### 3.2 Natural Language Processing Model

Concerns: The natural language processing model will be used to determine whether tweets are relevant to the cybersecurity field or not. The technical challenge in this is accurately teaching the model what categorizes relevance to the cybersecurity field. Additionally, a further challenge is in teaching the model in such a way that it doesn't become too specialized e.g. recognizes only particular subfields of cybersecurity as relevant but leaving out other important areas. It is extremely important that the model can recognize the context a word is used in as relevant tweets will contain many technical terms that are used in a different way in everyday life

Approach: We will be finetuning the BERT model to fit the classification task of deciding if tweets are relevant or irrelevant. BERT is a pre-trained natural language processing model put out by google. It was trained on exorbitantly large amounts of data (such as the entirety of wikipedia) that has allowed it to learn to represent words based on the context of sentences. This is very useful to us as its ability to represent technical terms is very important to our problem. BERT by itself merely produces a classification token (which is a vector representation for the inputted text in its entirety)

as well as vector representations for every word in the input text. To fit this to our classification task a fully connected layer will be placed on top of the BERT module. The fully connected layer will take in the classification token outputted by BERT and output either a 0 or a 1 representing whether the tweet is relevant or irrelevant. Once this architecture is implemented, the BERT+fully connected layer model will be finetuned on the dataset created during the data curation process to teach the model what is relevant to the cybersecurity field. Finetuning is done by training the entire model on the dataset in the normal manner, but only for a couple epochs.

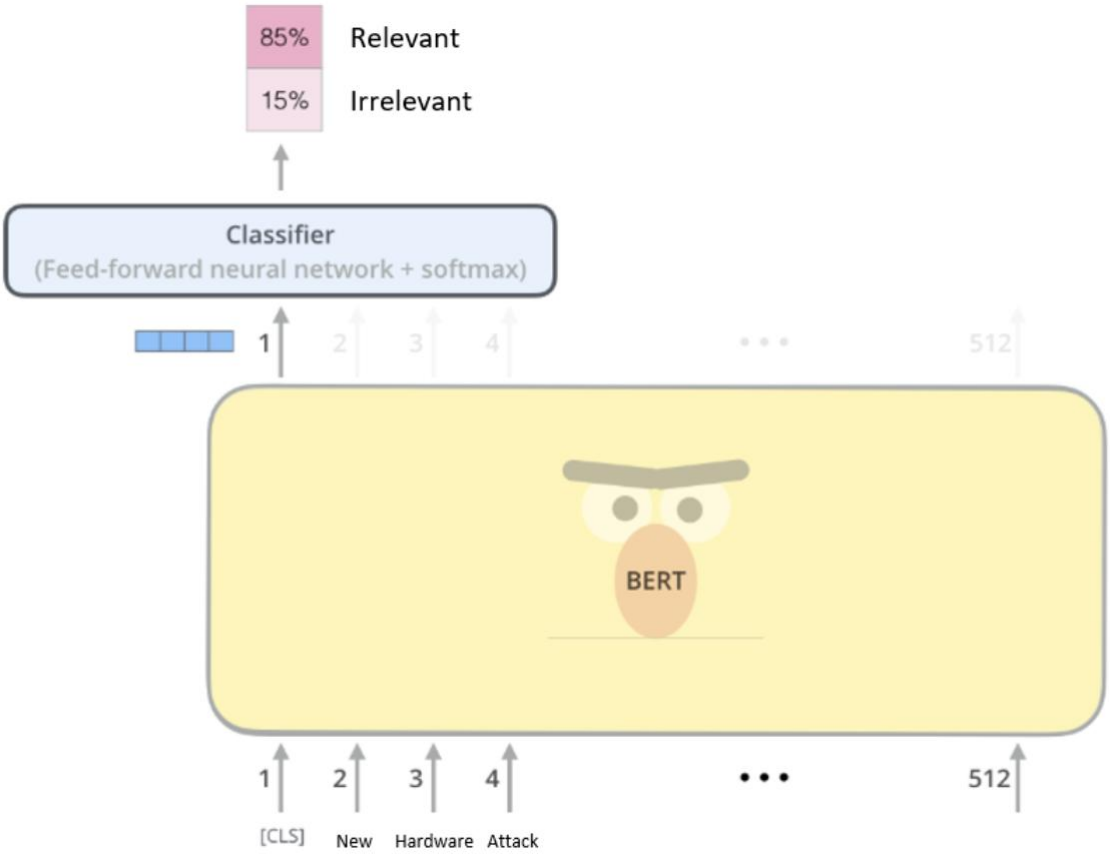



Fig. 4. Visualization of NLP model. Final model is BERT module pre-trained by Google with a fully connected layer on top that takes in CLS token outputted by BERT. Entire model is trained for a couple epochs to finetune BERT module to fit the cybersecurity problem

### 3.3 Noun Extractor, Clustering and Graph Analytics

Concerns: The Tweets reported back as relevant may have some relevant topics in its body such as the name of an IOC, hashes related to the IOC. Some Tweets might not have this but we will be required to maintain consistency in our database. We are also required to return these Tweets clustered based on certain attributes such as the IOC, Campaign etc. Our main challenge will be creating a rule based system for the former that can cover variation in tweet styles and a machine learning model for the latter to custer them into relevant buckets. The graph database must for relations on tweets with unique nouns and have accurate count for nouns encountered. Relations must also be formed accurately and in sustainable amounts to reduce storage constraints.

Approach: We plan to use Stanford's Natural Language Toolkit (NLTK) that will scrape the web page and return its contents. A rule based engine will be used for checking for relevant information (such as IOC, hash, threat campaign) from a URL as these are typically presented as nouns. We will be using multiple expressions for finding this in the document and have certain words such as common verbs, articles excluded from being relevant information. There will be another model that will be used for reducing the count of relevant words within a tweet. The tf-idf statistical model, short form for term frequency-inverse document frequency, will be used for finding words that are nouns but irrelevant. The tf-idf model will be trained on our relevant tweets dataset. We will also experiment between using a LSTM and a Transformer for this to experiment better methods for this problem. All our results from this will be stored in our graph database to form revelations between nouns encountered in the same tweet and grow these relations as nouns from other tweets are encountered. Authors making the tweets will also have nodes with relations to the nouns encountered in their tweet storing time of tweet. Time will uniquely identify tweets made by authors and can be used to detect bot accounts.

### 3.4 User Interface

Concerns: The user needs to be able to see what threat campaigns, indicators of compromise, and related tweets the system has found. Furthermore, all information the system finds has to be stored in such a way that it easy to retrieve it.

Approach: To address these concerns we will be using Neo4j's Graph Visualizer. With this we can view all nodes (Nouns and Authors) retrieved from tweets, relations between Tweets, Authors and find emerging threats from the database. This will be done through a terminal based UI for giving relevant tweets and Neo4j's Graph visualizer for looking at the nodes and relationships.

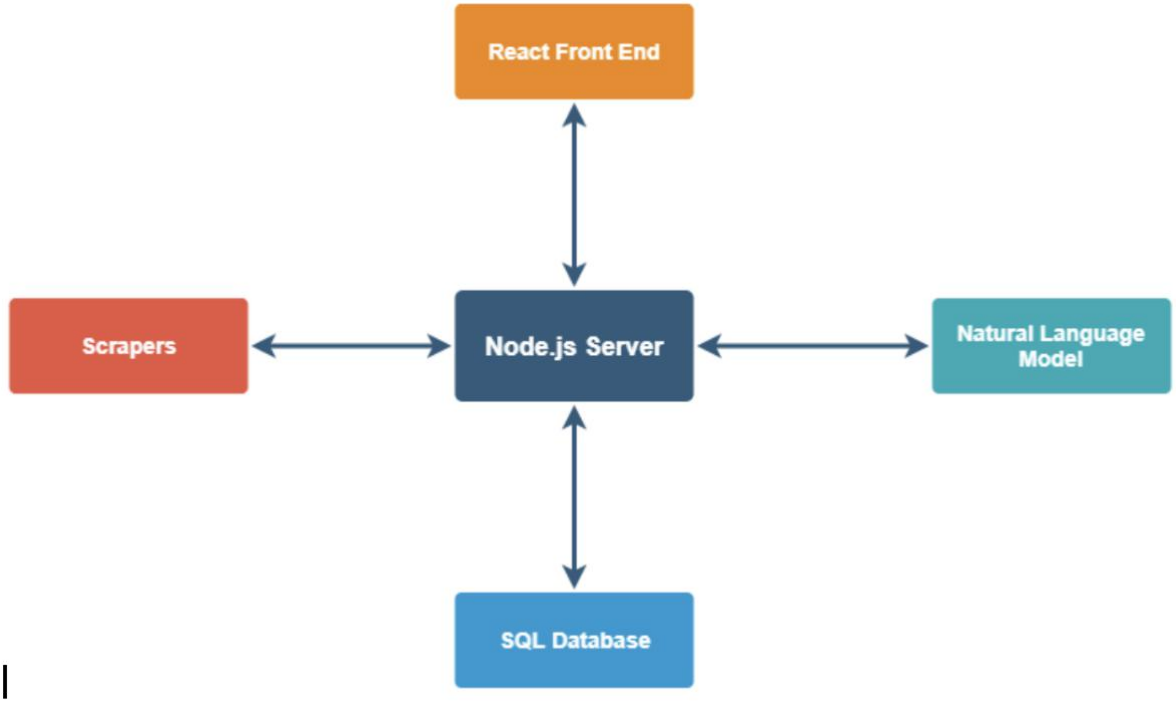


Fig. 5. Visualization of website. Server created in Node.js facilitates data transfer between scrapers, models, React front end, and SQL database

## REFERENCES


[1] Harshall Lamba. One Shot Learning with Siamese Networks using Keras. [online].
Available: https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras17f34e75bb3d

[2] Jacob Devlin Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing. [online].
Available: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

[3] Adrien Sieg Text Similarities : Estimate the degree of similarity between two texts. [online].
Available: https://medium.com/@adriensieg/text-similarities-da019229c894

### 3.2 Change Table

1. Removed UI from System Components and Requirements 2. Removed Web Scraper and Clustering 3. Added Noun Extractor, Clustering and Graph Analytics 4. Modified Algorithmic

## 4 Weekly Blog posts for Fall and Winter

### 4.1 Nikhil Anand

#### *4.1.1 October 17, 2019*

My name is Nikhil Anand I am part of the AI Based Threat Intelligence capstone group. This week we compiled all our problem statements and created a final submission, we also began looking at the format for our requirements document and are meeting with our client this Friday afternoon to discuss what exactly they expect out of us. The only problem we had this week was that our client was only able to meet with us Friday afternoon making it difficult to progress on our requirement document this week. Our plan for the next week is to meet as a group and finish the requirement document and get it approved by our client, and then submit it for this class.

#### *4.1.2 October 25, 2019*

This week we met with our client to discuss our first draft of our problem statement, and to get more specifics on what they wanted to see. Our group finalized our draft 2 for the requirement document. One problem our team faced was members coming in late for client meetings due to midterms, but this was a one time problem because our recurring meetings have been scheduled for a more accommodating time. Our plan is to discuss our draft 2 with our clients and get feedback, also we finalized our gantt chart, which is something our client was specifically interested in, and getting their feedback and modifying it with their thoughts will be useful for guiding our progress over this term, and the rest of the year.

#### *4.1.3 November 1, 2019*

This week the major problem that our group (Group 4 AI Threat Research) faced with this week is that the scope of our problem was drastically increased after talking with our clients about our specific requirements. We initially thought that all we had to do was classify tweets as relevant or irrelevant and produce the tweets to the clients. Now we have learned that we are being asked to take relevant tweets scrape any links in the tweets and try to classify IOC's that we will use to identify particular threats that we will try to cluster the tweets on. The progress our group made was that we shifted our research plans and the components of our project to include this new scope of the project. We also adjusted our gantt chart and worked on doing research this week. The plan for our group now is to complete our research and written documents by Wednesday next week, and we will share our research with each other to make sure every group member is on the same page. We will also try to have our initial twitter scraping tool done within two weeks. This tool needs to use the Twitter API to pull tweets and put them in a database. The uniqueness of this scraper is that we need to be able to control which queries we use to pull the specific tweets so that they can be relevant.

#### *4.1.4 November 8, 2019*

This week was spent primarily individually focused on the tech review. Each team member spent the week researching the technologies that may be used in each aspect of the project. Before this process started we met up and decided which roles each member would ”be in charge of” which was useful because it provides us with an ownership aspect

of the project. I personally was tasked with data storage, data procurement, and computational resources for training the model. No problems were encountered this week besides adapting our team roles to include the new scope of the project which we were informed about Thursday of last week. Our upcoming plan is to meet discuss the decisions and findings that we each made during our tech review and then begin working on part of our project which is the data procurement using Twint to build an automated tweet scraper and store the tweets which will later be manually labeled. We will also meet with our client next week to discuss our research and get their input.

#### *4.1.5 November 15, 2019*

This week the only progress we made was communicating our technology decisions to our clients, and set up a meeting for Friday evening to discuss some of the choices we made and, make sure we are all on the same page. There we no problems this week as we completed our technology review last week, and started discussing as a team our design document but have yet to make much progress on the actual document. Over the next week we plan on splitting up the document and completing our individual parts, then meeting up and editing and reviewing the document together.

#### *4.1.6 November 22, 2019*

This week our group made a lot of progress on the design document and were able to have a call with our clients to get feedback on our document. We will be completing the document on Saturday and turning it in. There wasn't many problems this week besides the fact that it was difficult to all work on the document this week due to other large assignments that team members had as well as interviews that were taking up team member time. We were able to push past it and get the work done, and the extended due date was very helpful. Next week we plan working on the scraping tool and figuring out how we will be using a tweet scraping tool for the final product as well, that is able to grab tweets in real time, which is a problem that was discussed during our initial meeting with our clients this week.

#### *4.1.7 November 29, 2019*

This week didn't make much progress due to Thanksgiving Break. We continued with some research but besides that no progress made. We did decide to shift from BERT to a different model which we will explore more next week. No problems were faced this week, besides the lack of work due to the break. Next week we plan on building the twitter scraper.

#### *4.1.8 January 10, 2020*

This week was a very productive week because we were able to mesh together the individual pieces of work that each team member worked on during the Winter break. We were able to get our data curation scraper up and running hooked up with the database and it now allows us to begin labeling tweets to prepare our training data for the model. A problem we had was that we are worried that we will likely get an imbalance of relevant vs. irrelevant tweets and are working to figure out a solution to help even out our numbers because good training data is crucial for allowing our model to work correctly and accurately. Our plans for the next week is to start labeling data and try to figure out a plan for how many tweets each member should work on and talking with out client to help brainstorm ways to make sure the irrelevant tweets don't overshadow the relevant ones.

#### 4.1.9 January 17, 2020

This week we made a lot of progress we were finally able to release the data curation web application that allows us to pull tweets from our database and then we manually classify the tweets as relevant or irrelevant and it gets stored in our classified tweets table which we will then be feeding into our BERT model for training it. The only problems we had was that we were having some issues deciding whether or not we should host the web application on the engr server or for each person to manually host it on their laptop when they want to classify tweets. Overall I made the decision that every person start the server on their laptop with one simple command ”npm start” because this gives the user more control over the application and can restart the server incase anything goes wrong. Another issue we were seeing was that emojis showed up as very weird characters in the database so there needs to be a script in place to ensure that only utf-8 characters are allowed in the database otherwise it could confuse the BERT model. The plan is for our team collectively to start classifying training data, we have set a goal of 1000 tweets by next week when we meet with our clients.

#### 4.1.10 January 24, 2020

This week we began labeling our data, and our goal is to have over one thousand labeled tweets done by Monday. We had to move this goal back a little from last week because of some technical difficulties but we are now back on track and have all began labeling the data. It is going by quite quickly and we should be able to exceed our expectations of one thousand tweets by then. We have added a counter on the side of our labeling tool which gives live counts of the number of relevant/irrelevant tweets which is nice. We were also able to get Bert set up and finalized the pipeline and have even began figuring out how we will be doing the fine tuning on top of the Bert as a service that we have implemented. There weren’t many problems this week, just that we added some new columns to the database and that wasn’t communicated properly leading to our tool not working correctly, but once that problem was realized we quickly solved it and got back to business. Our plans as we stated was to get at least one thousand labeled tweets done by next Monday and then begin training our model and figuring out how best to fine tune it to get the best results.

#### 4.1.11 February 7, 2020

This week we started testing the model with our dataset of approximately 4 thousand tweets (and counting). We were able to get accuracy of around 90 percent which is quite high! We made a lot of progress for our pipeline and demoed our project to our clients and they were pleased with our progress and encouraged us to continue working on the next (optional) steps that they wanted us to work on, which is the graph db and web crawling. The problem that we are seeing is that our data that we are testing the model on is quite ”related” to each other which adds bias in our testing, I believe to truly figure out our accuracy we need to find fresh tweets that are from completely different sources. I anticipate the accuracy dropping a bit but not too much so this isn’t a huge problem. We must now figure out our timeline for the new milestones that the clients want to see (graph db and web crawling) and email it to them. We will also need to research the new milestones and figure out what our game plan will be for completing it. Overall I am pleased with our progress and believe we are in a good position to present for Alpha Testing.

#### 4.1.12 February 14, 2020

The progress we made this week was fine tuning the model, which is an ongoing step because we will continue to be growing our data, so fine tuning will become more crucial as we diversify our data set more. We also have been working

on the draft of the poster and expect to be done Friday evening. There was no problems that we encountered this week. After completing the poster draft we will need to begin working on our presentation that is due next Tuesday, and the additional features that we would like to get to for the clients. This includes the graph database and web crawling/IOC extraction steps.

#### *4.1.13 February 21, 2020*

The progress we made this week was mostly on fine tuning the model, and preparing reports, flowcharts and diagrams for our design review. We also began more research on our graph database and started setting up Neo4J. The labeling tool UI was also fixed so that it was in a ready state for the design review. There were very few problems that were encountered this week besides the time crunch of preparing for the presentation and studying for a midterm that all three of us had on Tuesday. Now that we have completed the design review, our next goal is to start setting up the graph database and seeing how we can use the Stanford NLTK to create relations within it. We will also be working on the web crawling and more fine tuning as well.

#### *4.1.14 February 28, 2020*

This week we made progress on fine tuning the model and the graphDB which uses the NLTK noun extractor. The problems this week were mostly with the fine tuning, as it was difficult to gain access to a gpu that could do the fine tuning we needed. Next week we hope to work more on our pipeline and have a presentation for our clients as they are saying that they would like us to demo our project at some point.

#### *4.1.15 March 6, 2020*

This week we worked on the model as well as the graph database. In terms of the model the progress we made was creating a method for us to input our own tweets manually and having it instantly classify them, this was a useful tool both for demoing the model to our clients but also for us to learn how the model has been trained and where its strengths and weaknesses are. In terms of the graph db we worked on the script and were able to create a rudimentary pipeline for inputting tweets to create relations on different words from the nltk, but it currently only works on one word at a time we need to automate it so it works on the whole list. We are also looking into a different tool kit for extracting nouns from the tweets that may be more effective than our current tool. The problems this week was that our client expected a demo when not much has changed since our demo 2 weeks ago in terms of presentable items. We were able to communicate with the client our concerns and have moved the presentation later down the line when we have more progress made on the extra features we are working on. Next week we want to complete the graph db pipeline and start creating a final pipeline with all the features put together from start to finish.

#### *4.1.16 March 13, 2020*

This week we created our pipeline for our graphDB and fine tuned our model and tested it using datasets provided by our clients, and had success with our new testing datasets. We also updated our posted and turned it in. The only problems we faced this week was that it was hard to meet due to the COVID-19 thus most our work has been remote, and recording for our video will be difficult. We plan on making our video only screen-recordings, with voice overs since we can't meet in person. We plan on finishing our video this weekend and finishing our final report. Over spring break we will try and complete our final pipeline and communicate with our clients for feedback.

## 4.2 Rohan Varma

### *4.2.1 October 17, 2019*

My name is Rohan Varma and I am part of the AI threat intelligence group. This week we worked on our group problem statement and began developing questions to ask our client concerning our requirement document. The problem we faced was that we were unable to meet with our client until Friday afternoon. We will spend the next week crafting and finalizing our requirements document and also performing research into the best tools to implement our requirements.

### *4.2.2 October 25, 2019*

This week we met with our client and discussed what our end goals should look like. We were asked to prepare a plan detailing our visions for the project and deliver it to them by next week. We also worked on our requirements document and submitted it in accordance with the assignment. We will likely adapt our requirements document into the plan the client has asked us to prepare. In terms of problems, we had group members showing up late to meetings but everyone did their fair share of work so it didn't spiral into anything bigger. As far as plans, in our Gantt chart, we created some timelines for things to research in the coming weeks (compute power available to us, natural language processing techniques, etc.) and we will begin our investigations accordingly.

### *4.2.3 November 1, 2019*

This week we had another meeting with our client to discuss the project. We were faced with a large challenge as it turns out the scope of the project is significantly greater than we first thought. Accordingly, we have planned to update our requirements document and send our client a new project description to finalize the scope of the project and what we are to produce.

### *4.2.4 November 8, 2019*

This week I primarily worked on my tech review and completed the final draft. The biggest problem I faced was that I wasn't very knowledgeable about the portion of the project that I am becoming the lead for so I had to devote a lot of hours into getting up to speed on the technical details. Moving forward, I will begin working with my group on the design document as well as begin work on our scraping tool in accordance with our Gantt chart.

### *4.2.5 November 22, 2019*

This week we worked on our design document and had our weekly meeting with our client. The largest challenge we faced was again making sure our client was on the same page as us. They are pretty hands on in terms of the architecture and designs they want to see from us so we will spend the next week or two aligning our specifications with their criteria as well as continuing work on our scraping tools.

### *4.2.6 November 29, 2019*

The problem was that last week the client wanted us to have more specifics on how exactly the machine learning components of the system would work. Accordingly, as the machine learning lead, I performed more research into the machine learning architectures that are relevant to our project. I plan to begin prototyping some of these over the next few weeks.

#### 4.2.7 January 10, 2020

We continued work on our data curation process and have almost completed the tool that will allow us to label data for our training dataset. We encountered some issues regarding database access concurrency for our labeling tool but have devised a solution. We will implement this solution next week and begin labeling data.

#### 4.2.8 January 17, 2020

This week we got our labeling tool running and are beginning the labeling process. I am also putting up a bert-as-a-service server locally on my machine and I plan to put it onto an AWS instance. One problem we faced is that our client had questions about BERT, so we decided to send him articles detailing how BERT works.

#### 4.2.9 January 24, 2020

This week we began labeling data. I got bert as a service running on tweets. The problem I'm running into is figuring out how to deploy the system on aws and link it to our website. I will spend the next week figuring this out.

#### 4.2.10 February 7, 2020

This week we made tremendous progress. I got the BERT model pipeline working. This entails taking in json files containing tweet information, preprocessing them to remove empty tweets and non-ASCII characters, sending the results to bert-as-a-service to get tweet encodings, and then putting the encodings through either logistic regression or support vector machine layers to identify relevant tweets. A challenge I faced while putting all this together was that the different steps in the pipeline required different package versions, so I had to do a lot of trial and error in order to iron out the incompatibilities. Next week I will be further testing the efficacy of our pipeline on a new test set (our current test set was the result of splitting the tweets in the database so there was some relation between training and testing data) and finetuning the BERT module. I also hope to deploy the different parts of the pipeline in docker containers and write glue code that allows us to go through the entire pipeline at a click of a button.

#### 4.2.11 February 14, 2020

Progress This week I worked on our poster draft as well as finetuning the BERT model and testing the current model on a more general test dataset. I gained access to GPU servers and am beginning to test my modifications to Google's open-source BERT model to adapt their finetuning script to work for our tweet classification task.

Problems I did not have a GPU server for the majority of the week, which was a blocking point for my finetuning of the model

Plans We are going to prepare our presentation for the design review next week and I will continue the finetuning process.

#### 4.2.12 February 21, 2020

This week went into practicing the presentation and writing glue code for the various components of our project, specifically combining bert as a service with the classification layer in scikit learn. Challenges were the same package incompatibilities mentioned before but in this case figuring out how to write glue code around them was especially difficult. Next week I plan to finish writing the glue code and run the first tests with the finetuned model. We also plan to have a checkin with the client on next monday.

#### *4.2.13 February 28, 2020*

This week I focused on getting the client information they requested about the model's performance on the general test set. The largest challenge we faced was putting together a test set of acceptable size within a week as we needed to build a labelling tool. I plan to finish compiling the report they want and roll out the finetuned model this next week.

#### *4.2.14 March 6, 2020*

This week I developed tools to better demo our model to the client. I ultimately created an interactive tool that allows the user to enter in any sentence and then receive the model's prediction. The client also sent a test set that they wanted us to test the model on. This proved to be a significant challenge as the format they sent the data in was quite difficult to work with and required a lot of preprocessing. They also sent the data set one day before the demo call on Thursday which resulted in a lot of frantic work. Ultimately, we were able to perform some tests on the test set the client sent but not all of it. Accordingly, next week I will gather all the information the client requested about the model's performance on their test set. Additionally, we will be refining the training dataset to adjust the model's performance on sentences that are cybersecurity-related but deemed irrelevant by the client - e.g. ”register for our cybersecurity seminar on ransomware tomorrow!”

#### *4.2.15 March 13, 2020*

As I specified last week, I worked on getting the client all the information they wanted about the performance of our model. I compiled a 5 page report and sent it to thrm. We faced some difficulties due to some reprocessing requirements the client wanted carried out via some scripts they sent, but eventually we were able to get their pipeline working. Next week, we will focus on getting together our beta functionality video and our end of term progress report.

### 4.3 Harshvardhan Singh

#### *4.3.1 October 25, 2019*

The group got together this week to have a concrete plan on what to do in the project and by when. We completed the requirements document and had a meeting with our client. We will be presenting our plan to our client this Wednesday/Thursday and have a recurring series set up to meet them. We have a concrete plan defined in our GANTT chart and feel that sticking to that will ensure success in this project. Going forward we will be working on looking at how to use the Twitter API.

#### *4.3.2 November 1, 2019*

We had a talk with our client and got to know more about our project. It turned out we were missing half the goals of our project but was a very informative call. Our main problem for the week was coming up with strategies for the new portion of our assignment. While we haven't exactly planned how to tackle the new problems, we are actively researching ideas on this. Our main goal is to research ways for scraping pages to get relevant ideas. We are also modifying our GANTT chart to accommodate for this and are recreating our requirements document. Apart from this I am mainly researching new web scrapping APIs and the Twitter. The goal is to come up with a quick test on how the Twitter API works by the end of this week.

#### 4.3.3 November 8, 2019

This week I primarily worked on researching the technology that we will be using for the assignment. This part was very time consuming as the research for this was very far spanning, based on parts of the system that others are working on defining in this document. Over all, I feel that working on the requirements cemented a lot of the parts of the system that I was very confused about. The main problem was comparing the different technologies and choosing the best ones amongst them. I felt that my choices for choosing some technologies was subjective as I getting the license for them might be hard and even time consuming. Even for the ML algorithms, I had to chose based on what algorithm is known to be marginally better in most cases but couldn't actually compare the results as we needed the data set created before that. The plan for the coming week is for the group is to come together and have a talk about all the technologies we will be using

#### 4.3.4 November 15, 2019

This week I was in contact with the client and discussed why we chose to do things in a certain manner. We have a meeting scheduled for Monday to discuss this. I feel that we had many assumptions on the system which I specified in my email to them. Overall not many problems this week. Mainly had conversations in emails for which I had to research different system parts and talk about why we were using them. The plan for the coming week is to have a meeting with the client and make sure we all are on the same page. I feel we are at a point in our capstone where we are very close to getting everything down to the last detail. Apart from that, the group will be meeting to work on the group assignment due this coming week.

#### 4.3.5 November 22, 2019

We all got together to write our design document. Apart from that, we had a meeting with our client and got a lot more clarity on how the system is supposed to perform. I drew a flow chart for our system which made all the parts of the system a lot more concrete for me. It gave me an understanding into what could be redundant. The main problem for this week was drawing the flow chart. This required me to review the basics of chart notations and related stuff. Drawing the actual system was also challenging as it required all the parts about it to be concrete. Apart from this, the design document's initial requirements were kind of hard to understand but this made a lot more sense after meeting my TA. Upcoming plans are to draw a DB schema and send it to our client. Proofreading and fixing errors in the design document are also something we have to do. We plan to send our client the completed document by the end of this week and will need to work on making sure everything is concrete and there are no inconsistencies in our document.

#### 4.3.6 November 29, 2019

We didn't have a lot of progress this week mainly because of thanksgiving. The main problem was not being able to have a final meeting about the final draft with everyone. We will be sending the draft this weekend with minor changes to have it signed of.

#### 4.3.7 February 8,2020

We had a demo with our client for our current system and they seem to be very impressed with our progress. We have to finish the graph db followed by the web scraper to complete our capstone. I am working on the graph db and am trying to split the work with others in our group. Not everybody seems interested in working on more stuff. The plan for the week is to set the graph db and migrate everything from mysql to neo4j and continue labeling data.

#### 4.3.8 February 14, 2020

I started learning the Cypher programming language and have started forming relations with simple data types with it. Our progress on labeling has slowed down a bit but we plan to get back on it after the capstone presentation. The main problems were just syntax related as we are learning the language at the moment No major blocks have occurred this week. The plan for the coming week is to continue labeling data, developing the graph db after we are done with the capstone presentation due to midterms and assignments being due on the same day too. The graph DB should be up by the end of next week or mid-week after the coming week.

#### 4.3.9 February 21, 2020

We finished presenting our presentation to Kirsten and felt it went very well. We spent most of our initial time creating a presentation for the meeting and have been working on finishing up on the graph database. The problems for this week has been dedicating time to work on the graph db. With the presentation out of the way now, I am trying to coordinate different members to work on the graph db and finetuning the ML model we have. I myself am also working on creating queries for the graph db. The plan for this week is to finish work on the graph db. We have a meeting with our client on Monday to talk about our project status and talk more about what we have done and what we need to do to finish the project.

#### 4.3.10 February 28, 2020

We worked on classifying more data to create a document on how our classifier works on new data. We curated our dataset, manually labeling around a 1000 tweets and applying our ML model on it. Our accuracy went down a bit but we are happy as tweets in this case were wildly different from our previously labeled tweets. Apart from that I have been working on getting the graph database up. I have tweets loaded into the graph database but need to start forming relations on it which should be done by the end of the week. The main problem for the week was labeling a 1000 tweets in a few days as our client wanted us to give them some more concrete statistics on how our model is performing. Also, working with documentation for neo4j's python API has been challenging. The plan is to wrap up the graph database by the coming week and focus on delivering the system as a whole with fully integrated parts. After that we plan to increase the model accuracy by labeling more tweets and retraining our model.

#### 4.3.11 March 4, 2020

We created a few relations with the graph db and have the python API up for the same. We continue to label tweets to make our model better. We have a cmd line tool ready to demo the performance of our model which our clients are expecting on Thursday. We didn't encounter any major problems this week. We are trying to wrap up all our work and create a working pipeline for the system now. Plans for the coming week are talking to our client and recommending changes to the system (Switching from the graph db to elastic search) and working on completing the pipeline.

#### 4.3.12 March 13, 2020

We have been working on wrapping up the system this week. I could not work a lot this week as I've been out with the Flu. No problems encountered so far. We have been trying to wrap up other capstone related assignments such as the demo video and the poster.The plan for the coming week is to finish all upcoming assignments and then modify my script for the graph database to form relations with all keywords rather than the one keyword that I hardcoded it to do.

# 5 Tech Reviews

## 5.1 Nikhil Anand

CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 9, 2019

# AI THREAT INTELLIGENCE

PREPARED FOR

MCAFEE LLC

PREPARED BY

GROUP 4
AUTOMATED RESEARCH TEAM
NIKHIL ANAND

**Abstract**

The problem that we are trying to solve is that currently researchers and engineers at McAfee are using precious work hours to scour the web for new and relevant threat campaigns, malware strains, and other cyber security related events. Once they have identified which threats are relevant and worth prioritizing they begin to understand the threats and reverse engineer them to make sure that McAfee is able to identify and protect against these threats in the best ways possible. My role throughout this project will be to manage our cloud resource needs, which includes data storage and the cloud compute power. I will also be in charge of the Tweet Scraper which will be used in the data curation process of the project. This technology review will discuss the various options that were assessed, and the rationale behind the choices that our team made.

# CONTENTS

**1 My Role** 2

**2 Responsibilities** 2

2.1 Database . . . 2

2.1.1 Relational vs. Non-Relational Database . . . 2

2.1.2 Amazon Web Services vs. Google Cloud Platform . . . 3

2.2 Computational Resources . . . 3

2.2.1 Amazon SageMaker vs. Cloud Machine Learning Engine . . . 4

2.3 Scraping Tools . . . 4

2.3.1 Twitter API vs. Twint . . . 4

**3 Technology Decisions** 5

3.1 Database . . . 5

3.2 Computational Resources . . . 5

3.3 Scraping Tools . . . 5

**References** 5

# 1 My Role

The various tasks required to accomplish the goals for this project have been divided amongst the team members of this project. The purpose of this document will outline in detail the three roles that I have been tasked with, as well as the technologies and decisions we have reviewed in order to move forward with the project. The three roles that will be discussed later are data storage which includes deciding the database architecture, and the cloud provider which will be used to host the database. The second role is deciding which cloud platform to use to train our model with, since a local machines compute power is not enough for this part of the process. My final responsibility is preparing the tweet scraper, for this process we will be looking at the optimal methods for acquiring tweets to be used in our datasets.

# 2 Responsibilities

## 2.1 Database

For this project we will be using a database for a variety of reasons. The first reason is for storing the training data and testing data along with their respective labels. After the model is created, we will also be using the database to store relevant tweets, various indicators of compromise, and finally a table of unique threat campaigns that have been identified. For this aspect of the project, we will need to decide whether to use a relational or non-relational database as well as which platform to host our database on.

### *2.1.1 Relational vs. Non-Relational Database*

Relational databases store data in tables and rows. Relational databases are based on an aspect of set theory called relational algebra [1]. This essentially means that you are able to utilize functionality such as unions, intersections, set-differences, compliments, disjunctions, inner joins, and outer joins as well as many other useful operations that originate from relational algebra. Relational databases use Structured Querying Language (SQL) which make it a very safe and reliable database system to utilize. SQL has 4 core properties known as the ACID properties which ensure data integrity during transactions [3]. ACID is an acronym for Atomicity, Consistency, Isolation, and Durability. Atomicity is the property within SQL which ensures that all operations including inserts, updates, and deletes inside of a transaction either complete successfully or are rolled back to their initial state before the transaction took place. Consistency within SQL refers to the promise that any failures, errors, or problems that occur during any stage of a transaction will never be passed on to the database itself, only a complete success will be passed on to a database. Isolation is the third property of SQL, this aspect refers to the fact that every transaction occurs in isolation, and no transaction can access the result of another until the transaction is fully complete. Isolation also ensures that no operation can be run by multiple transactions concurrently. And finally, Durability, which states that once a change has been made to a database, that change is permanently committed in the system, regardless of any failures, abnormalities, or crashes [3]. The structure of relational databases are unique because it allows for linking between various tables with the use of foreign keys, which essentially are used as pointers to information. This aspect is extremely important for relational databases because it prevents data replication and allows for heavy applications to be supported.

While relational databases store data in tables and rows, non-relational databases store data in collections of documents [2]. Typically documents store data in for the form of JavaScript Object Notation (JSON) which allows for data to be stored using key value stores. One important aspect is that non-relational databases allow for much more flexibility for storing information. The benefits of non-relational databases are that they can process any type of data

without needing to modify the database architecture [4]. This makes creating and modifying data much quicker and cheaper than in a relational database.

#### *2.1.2 Amazon Web Services vs. Google Cloud Platform*

Amazon's Relational Database Service (RDS) is the current market leader in cloud relational database services. It supports six major database engines including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server. The standard RDS provides a maximum of 6TB storage, and can be scaled virtually through the AWS console. RDS has great functionality in terms of security and recovery, utilizing storage volume snapshots, RDS is able to allow users to revert the state of the database back to any point-in-time snapshot. RDS can also be launched using Amazon VPC which allows deployments in private networks. Amazon also provides industry leading monitoring and metrics included with the standard RDS package. Along with this they also provide automatic database patches and backups [5].

Google's cloud database service is called Cloud SQL, it is the counterpart to Amazon's RDS, but unlike the variety of database engines supported by RDS, Cloud SQL only supports MySQL. Cloud SQL allows for a maximum storage capacity of 10TB and allows users to manually configure automatic scaling in the instance settings. Google's Cloud SQL is recommended to new users, as it is extremely easy to deploy and manage [5]. An important concept to note is that neither AWS nor Google Cloud allow for very easy cross platform traffic, meaning whichever provider is used for the computational resources should be utilized for the data storage as well.

### 2.2 Computational Resources

A core element of this project will be creating and training a model that is able to classify tweets as either relevant or irrelevant. For this to be possible, we must be able to test the model many times with large datasets of tweets in order for it to learn effectively. Even simple deep learning models can take days to train if they are run on laptops which generally don't have graphic processing units (GPU's) to help speed up the process. During the training process of a deep learning model, the two main operations that are performed are the forward pass and the backward pass [6]. In a forward pass, an input is passed through all the layers in a model, resulting in an output. While in a backwards pass, we use the output that is generated to figure out our error also known as loss in the model, and we propagate backwards through the model, updating the weight matrix of each layer within the neural network. To put in perspective how taxing of an operation training these models can be, we will examine VGG16 as an example. VGG16 is a convolutional neural network, with 16 hidden layers, and is very commonly used in deep learning applications. This model has about 140 million combined weights and biases all which must conduct various matrix multiplication operations. Training a simple model like this would take far too long on any traditional machine [6]. This example shows us exactly why we need to utilize different computational resources in order to effectively train our model. Using GPU's is important because the benefit of GPU's is that they have hundreds of simple cores that are able to operate concurrently and are able to perform matrix operations extremely quickly. The two main options that we are looking at with this project is Amazon SageMaker, which is Amazon's deep learning accelerator and Cloud Machine Learning Engine offered by Google. Both these providers are cloud providers, and are ones that McAfee consistently utilize in their day to day operations.

#### 2.2.1 Amazon SageMaker vs. Cloud Machine Learning Engine

Amazon SageMaker vs. Cloud Machine Learning Engine Deploying a machine learning service on cloud platforms is a great way to deploy computationally intensive machine learning models. The reason is because the heavy lifting is done by the serverless cloud engines that are able to utilize many of the top of the line processing tools to quickly and efficiently run the models and relay the information back to the users. Amazon SageMaker and Google's Cloud ML Engine are the two market leaders in cloud based machine learning platforms making either of them the obvious choice for this project. In order to differentiate them we will look at a variety of factors in which they vary slightly. The first factor we will look at is deployment. SageMaker allows for users to begin deployments with only one click when logged into the AWS console which makes deployment extremely intuitive and simple, whereas Google requires the setup of a notebook server, which is then launched using the Google cloud shell in order to begin the deployment. While both operations are doable, SageMaker's overall process is much cleaner and concise giving it the slight edge in this area. Another area where SageMaker edges out Google ML Engine is in its ability to deploy multiple variants of the same model using a provided HTTPS endpoint allowing for us to test a variety of modifications concurrently rather than deploying them one at a time on Google's ML Engine. This benefit will help us save time, and allow us to optimize our model in our allotted time frame. In terms of pricing both platforms are essentially equal as it the pricing based on usage and power allotted.

### 2.3 Scraping Tools

For this project we will be required to create our own dataset used for both training and testing the model. In order to create the dataset we will need to first procure many tweets that are related to cybersecurity. Then we must go through and label each tweet as either relevant or irrelevant. In order to speed up this process we will be creating a tool that collects tweets and store them in a database, and then labeling the tweets and linking the appropriate tweets with their labels. The decision that we need to make is to decide how we will be obtaining the tweets to be classified. There are two options that we believe would be appropriate for this, the first being the official Twitter API and the second being Twint, which is an advanced twitter scraping tool that doesn't utilize Twitters official API.

#### 2.3.1 Twitter API vs. Twint

1. Twitter API has three options, the standard option, premium, and finally their enterprise option. For this project we would want to focus on being cost effective, meaning we will focus on the standard API package which is primarily used for testing, integration, and concept validation. With the standard API we are able to create, retrieve, and delete tweets, retweets, and likes. For the scope of the project the ability to retrieve tweets is the only aspect that is relevant to us. The API also allows for tweet filtering using keywords, hashtags, and categories, this is very important because it will allow us to limit the tweets that we are pulling to be relatively close to what we are looking for, which will reduce overhead time of removing tweets that are completely unrelated to our topic of cybersecurity threats. Unfortunately, with the standard API you are only able to search tweets that were published within the last 7 days. This limitation is quite detrimental as it will be difficult to accumulate a large enough dataset of relevant security tweets within only a span of the last 7 days. Another drawback of the Twitter API is that the enterprise version will cost McAfee well over $2,500 a month, if we end up implementing the model with this tool.

Twint on the other hand is an advanced scraping tool created in Python to help users get around the stringent restrictions that the traditional Twitter API places on its users [7]. Twint uses search operators to collect tweets by topic,

trends, hashtags, and keywords. Twint is easily setup with starter code provided on the official Github page. There are no rate limitations, it is free, and it can access almost all tweets without the restrictive limit of 3,200 tweets per user which is placed by the official Twitter API.

## 3 Technology Decisions

### 3.1 Database

For our data storage needs we decided to go with a relational database architecture because we believe it would be the most efficient way to store the data that we are collecting. This makes sense because with a relational database we can create links between individual tweets, IOC's that may be related to them, and which threats each tweet is associated with. We also believe that hosting our database on Amazon's RDS would be our best choice because it meets all requirements for data storage capacity, security, and upkeep. Additionally, we will be utilizing Amazon for our computational needs and connecting our database services with another Amazon service is much more cost effective than having cross traffic between various cloud providers.

### 3.2 Computational Resources

For our computational needs we have decided to move forward with AWS SageMaker, for the primary reason that it is easy to utilize and setup as well as its ability to train multiple variants of the same model allowing for the tuning process to be more efficient. Both platforms would be able to handle our computational needs, but since we believe SageMaker is able to help speed up our process because of the features it provides, along with the fact that McAfee already utilizes this service, and Amazon's database services were a great fit, we felt that SageMaker was the best choice for us.

### 3.3 Scraping Tools

When choosing the correct tool to procure tweets, Twint seems like the obvious choice based on our research because not only does it allow us to collect tweets without a restrictive 7-day period, but it also provides us with an already built environment, and is well documented. Twint seems to offer almost all the same features as the Twitter Enterprise API which would cost McAfee over $2,500. For these reasons we will be moving forward with using Twint to procuring the tweets that will be used in our training and testing sets.

## References


[1] Vaidehi Joshi. *Set Theory the Method to Database Madness*. [online].
Available: https://medium.com/basecs/set-theory-the-method-to-database-madness-5ec4b4f05d79

[2] Jacqueline Homan . *Relational vs. non-relational databases: Which one is right for you?*. [online].
Available: https://www.pluralsight.com/blog/software-development/relational-non-relational-databases

[3] Suresh Basinasetty. *ACID Properties in SQL Server*. [online].
Available: https://www.tutorialgateway.org/acid-properties-in-sql-server/

[4] Microsoft Azure. *Non-relational data in NoSQL*. [online].
Available: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data

[5] Asaf Yigal. *Google Cloud vs AWS: Comparing the DBaaS Solutions*. [online].
Available: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data

[6] Faizan Shaikh. *Why are GPUs necessary for training Deep Learning models?*. [online].
Available: https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/

[7] Francesco Poldi. *Twint*. [online].
Available: https://github.com/twintproject/twint

### 5.2 Rohan Varma

Next Page

Oregon State University
College of Engineering

# CS Capstone Technology Review

November 8, 2019

# AI Threat Intelligence

Prepared for

McAfee

Prepared by

Group 777

ART

Rohan Varma

Signature Date

**Abstract**

I will be the machine learning lead for the project. As such, rather than pieces of the project, I am responsible for the phases of model development. The phases are model architecture selection, model implementation, and model evaluation and tuning. Research into possible technologies for executing these phases yielded the following candidates that are discussed in detail: Recurrent neural networks and long short term networks for model architecture; Keras and PyTorch for implementation frameworks; TensorBoard and matplotlib for evaluation tools.

# 1. Role

My role will be the machine learning lead for the project. I will be responsible for ensuring that the machine learning models we create perform to acceptable standards and meet the requirements we set. I will also oversee researching the technologies necessary to classify tweets as relevant or irrelevant. Based off my research, I will be responsible for choosing model architectures and guiding the team in implementing and training them. I will also be responsible for creating metrics for accurately evaluating the performance of our models.

# 2. High Level Objective

Our client is a cybersecurity company that devotes a lot of time into researching what the latest threats are. Our goal is to reduce how labor intensive this process is by automating the initial filtering of information on the internet. More specifically, we will be gathering tweets and blog posts that are relevant to developments in the cybersecurity field to point the client towards "indicators of compromise" that are worth investigating. Ultimately, our goal is to create a system that can streamline the client's research process by bringing relevant indicators of compromise directly to them.

# 3. Pieces of Project

As my role is to oversee the development of our machine learning models, my responsibilities are better classified as phases of a model development process rather than individual pieces. Accordingly, the phases of the model development process and their demands are described below.

## 3.1 Selecting Model Architectures

Determining whether tweets/blog posts are relevant will rely heavily on machine learning techniques. However, machine learning is a broad field, and there are many different model architectures out there. At a glance, it is quite apparent that the natural language processing subfield of machine learning is likely the most relevant area to draw architectures from. However, even this focus of machine learning has many different architectures to choose from. As implementing all of them is infeasible, deciding which architectures to implement and test will be the first step. Throughout this process, different architectures and preexisting implementations will be investigated in order to gauge which architectures are promising for our problem. Ultimately, the list of models we wish to implement will be finalized in preparation for the next phase

## 3.2 Implementing and Training Selected Models

After chosen models are selected, proprietary code will need to be developed in order to meet our client's standards. While preexisting models will provide a starting point, many changes will have to be made in order to tailor the models to our problem as well as to deploy them on the compute resources available to us. Once the models are implemented, monitoring their performance as they are training and tabulating this data is key. Code will have to be created for this data tabulation process independent of the model implementations.

## 3.3 Evaluation of Models and Tuning

A key part of training any machine learning model is evaluating its performance afterwards. While this may seem very straightforward at first, there are many nuances. Straightforward metrics such as accuracy are not always reflective of the full picture, and a deeper look into things like the number of false positives, false negatives and other metrics are also required – in our problem, it is far worse to have false negatives and miss out on important threats than it is to display some information that isn't as relevant. Additionally, after a model's performance is understood within the context of the problem at hand, the next step is deciding what changes need to be made to the model in order to boost its performance. These changes can be made either in terms of the architectural structure of the model or in the form of tuning the models various hyperparameters.

# 4. Possible Technologies

## 4.1 Architectures

This phase of the model development process is entirely research based. Accordingly, some research into relevant machine learning techniques and ultimately some model architectures of interest are listed below

### 4.1.1 Sentiment Analysis

As mentioned above, it is obvious that natural language processing is the specific sub area of machine learning that our problem of tweet and blog post classification falls into. However, further research has suggested that within natural language processing, our problem is a "sentiment analysis" problem. More specifically, sentiment analysis is described as "contextual mining of text which identifies and extracts subjective information in source material [1]." Essentially, sentiment analysis is a sub-area of natural language processing that encompasses extracting contextual meaning from text, which is exactly our scenario – our goal is to extract whether source material is relevant to the realm of cybersecurity. Accordingly, architectures suitable for sentiment analysis will be researched

#### 4.1.2 Recurrent Neural Networks

As stated by Andrej Karpathy, one of the leading authorities on deep learning architectures, "what makes recurrent neural networks (RNN's) so special" is that "they allow us to operate on sequences" rather than the fixed vectors vanilla and convolutional neural networks are limited to [2]. This is extremely useful to our sentiment analysis problem as we will be classifying sentences and paragraphs of varying lengths. Additionally, another significant advantage of RNN's is the fact that they can handle these varying lengths in a "fixed amount of computational steps [2]." Lastly, the most promising property of recurrent neural networks is their ability to use information from throughout the sentence rather than simply looking at one word in isolation at a time. This is done by processing a "hidden state" that keeps track of prior information/words [2].

#### 4.1.3 Long Short Term Memory RNN's

While one of the appeals of vanilla RNN's is that "they might be able to connect previous information to the present task", the truth is that they often to struggle with long-term dependencies [3]. In the context of our problem, this manifests itself in the following way: say for example, that the words at the end of a sentence depend on context from the very beginning of the sentence. As the distance between the start and end of the sentence grows, at some point vanilla RNN's "are unable to learn to connect the information [3]." Long Short Term Memory RNN's (LSTM's) take care of this problem and are a special type of RNN capable of learning long term dependencies. LSTM's gain this capability by introducing "forget gates", mathematical additions to the basic RNN unit that allow the network to assign more importance to information farther away when necessary [3]. On the flip side, adding these gates significantly increases the mathematical parameters that need to be trained, increasing the complexity of the model and in turn the difficulty of training it.

From this, it is clear there is a tradeoff between learning long term dependencies and model complexity. As a result, LSTM's should only be used when RNN's are not sufficient to handle the task. Relating to our problem, this means that RNN's should probably be used for tweets since they are limited to 140 characters whereas LSTM's should be used for blog posts, which are more likely to require learning long term dependencies.

### 4.2 Model Implementation and Training

#### 4.2.1 Keras/TensorFlow

As found on their documentation, Keras is a "high-level neural networks API, written in Python and capable of running on top of TensorFlow" that is maintained by Google [4]. It aims to provide "easy and fast prototyping" and supports both convolutional and recurrent neural networks [4]. Some other key aspects of it are that it is integrated with the rest of the Google machine learning environment, meaning that it fits seamlessly with cloud compute resources such as the Colaboratory as well as model evaluation software such as TensorBoard [4]. Its main limitation is the fact that it is high level, so creating custom models becomes more difficult.

#### 4.2.2 PyTorch

"PyTorch is an optimized tensor library for deep learning using GPUs and CPUs [5]." As such its base purpose is the same as Keras's. In fact, from the code samples found on their respective documentation sites, their code is often structured the same way for typical model architectures. PyTorch, however, has a stronger focus on customizability, and its API isn't as high level as Keras's [6]. The main area where this can be seen is the way models are trained – PyTorch requires the user to pencil out the training procedure in far more detail than Keras – which largely automates the process [6]. Accordingly, it is much easier to tweak the training procedures and model architectures to a custom process in PyTorch. On the flip side, PyTorch doesn't come with built in model evaluation tools like Keras [5].

Ultimately, we see that the main tradeoff between PyTorch and Keras is ease of use/user friendliness vs fine grain control over the models. While both frameworks have out of the box support for the standard versions of vanilla RNN's

and LSTM's, the extent to which we will have to modify the preexisting architectures is unclear. As such, decisions about which framework to use will be made once it is known whether Keras gives us enough fine grain control, allowing us to leverage its relative ease of use.

### 4.3 Model Evaluation and Tuning

#### 4.3.2 TensorBoard

As briefly mentioned above, TensorBoard is part of Google's machine learning suite and is used to "provide the visualization and tooling needed for machine learning experimentation [7]." Essentially, TensorBoard hooks into Keras and TensorFlow and produces automated graphs detailing the model's performance throughout training and on testing sets. It also creates graphs detailing the model's data flow process and compiles parameters values upon demand.

#### 4.3.3 Matplotlib

Whereas TensorBoard is an evaluation suite, matplotlib is merely a graphing/data visualization library [8]. As such, it does not automatically track and create graphs of model performance and data flow. However, all of this can be achieved by writing PyTorch code that compiles this data and then uses matplotlib to graph it. Although using matplotlib requires this extra layer of work, the main advantage it presents is that it is compatible with any machine learning framework unlike TensorBoard.

At the end of the day, the consideration between these two packages comes down to whether we can use Keras or not. If we go with Keras, then TensorBoard is far and away the better option as it saves us a lot of time and effort. However, if we use any framework outside of Google's suite, then matplotlib is our next best option.

## 5. References

[1] S. Gupta, "Sentiment Analysis: Concept, Analysis and Applications," *Medium*, 19-Jan-2018. [Online]. Available: https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17. [Accessed: 09-Nov-2019].

[2] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," *The Unreasonable Effectiveness of Recurrent Neural Networks*, 21-May-2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Accessed: 09-Nov-2019].

[3] C. Olah, "Understanding LSTM Networks," *Understanding LSTM Networks -- colah's blog*, 27-Aug-2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 09-Nov-2019].

[4] Google, "Keras: The Python Deep Learning library," *Home - Keras Documentation*. [Online]. Available: https://keras.io/. [Accessed: 09-Nov-2019].

[5] Facebook, "PyTorch documentation¶," *PyTorch documentation - PyTorch master documentation*. [Online]. Available: https://PyTorch.org/docs/stable/index.html. [Accessed: 09-Nov-2019].

[6] P. Migdal, "Keras or PyTorch as your first deep learning framework," *deepsense.ai*, 10-Sep-2019. [Online]. Available: https://deepsense.ai/keras-or-PyTorch/. [Accessed: 09-Nov-2019].

[7] Google, "TensorBoard : TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/TensorBoard. [Accessed: 09-Nov-2019].

[8] J. D. Hunter, "Installation¶," *Matplotlib*, 28-Oct-2019. [Online]. Available: https://matplotlib.org/. [Accessed: 09-Nov-2019].

### 5.3 Harshvardhan Singh

Next Page

# AI Threat Intelligence

Team ART (Automatic Research Team)

Harshvardhan Singh, Senior Software Engineering 1, Fall

***Abstract*— With advances in sharing information in different innovative ways, we are working on collecting Threat Intelligence using Twitter for McAfee. As many researchers talk about there research/findings on Twitter to their audience of domain field experts spread across the world, we seek to create a system that will find tweets relevant to researchers on cybersecurity incidents using the Twitter API and a Deep Learning Model. If the tweets returned have hyperlinks linking into other pages, the system will also mine those pages returning relevant information such as artifacts and Indicators of Compromise. We also plan to create a cloud hosted database that will store our dataset which we create semi-automatically.**

Table of Contents

1. **Role** ..... 3

2. **Technologies** ..... 3

2.1 Website Scraping API ..... 3
2.2 Clustering Information based on Relevant Features ..... 3
2.3 Tweet Relevance Machine Learning Algorithm ..... 4
2.4 Graph Database……..... 4

3. **References** ..... 4

## 1. ROLE

My role in this project is to create the parts of the system where our project will use an API to parse pages and report relevant information back to the user. I will also be working on creating the algorithm that will cluster information reported from this API into relevant pages. Clustering in this case will depend on threat campaigns names, similar incidents, geography of attacks and other features that we will identify. A threat campaign is any type of malware or security incident going around. My final part will be a part of the Machine Learning stage for classifying Tweets where I will be working on prototyping of different machine learning algorithm architectures perform on our dataset to create a base model to compare the final model to.

## 2. TECHNOLOGIES

### *2.1 Website Scraping API*

This part of the system is used when our Machine Learning Algorithm classifies a Tweet as relevant, and the Tweet has a link in it to another webpage. The API will look for relevant features in web pages. Relevant features can range from the name of the attack, IOC's (Indicator Of Compromise) to the geography where attacks are occurring. We will be returning the results to a database. The options for the API are Scrapestack, Webscraper and Parsehub. The advantage in using Scrapestack is that it is supported across multiple platforms such as Python and JavaScript and should enable me to manipulate the data and return it to the database without spending much effort into learning new languages. It also reports back results in seconds, has a very high uptime and handles a billion plus requests a month making this a mature API to use. The downside would be that we will need the basic tier for using this API to our needs, which costs around twenty dollars a month and handles two hundred thousand requests. This provides the best bang for buck at the moment as it is the cheapest API with all the features we need. WebScraper.io offer's an API that also allows us the same functionality as Scrapestack but at a higher cost of a hundred dollars a month and does not give us a fixed number for sites accessed. Sites accessed are based on credits that are allocated from WebScraper. While WebScraper has better documentation on certain features, they do not talk about the languages they support. Parsehub has the best documentation out of them all and all the features we need for our functionality. The only con with this is that it is the most expensive API listed above, with a monthly price of one hundred and forty-nine dollars. Valuing documentation, data retrieval and scalability as the deciding factors, I will be going ahead with Scrapestack as this is reasonably priced, returns data instantly and does a decent job on documentation for its API.

### *2.2 Clustering Information based on Relevant Features*

This part of the system is after we have all the data returned from our website scraping API. After we have our data retrieved, we will be clustering our tweets based on similarities between them. There are two aspects to clustering, one of them being giving the indicator of compromise and the other being checking how similar attacks are to each other. Another technique I will be employing is using Stanford's Natural Language Toolkit to find nouns out of a Tweet which generally hold important information. From this I will remove general nouns using tf-idf (term frequency-inverse document frequency).

#### *2.2.1*

For finding the IOC's I plan to use a regex expression to find different IOC's from a precompiled list of IOC's. Examples for IOC's are Unusual Outbound Network Traffic Anomalies in Privileged User Account Activity, Geographical Irregularities, Log-In Red Flags, Increases in Database Read Volume etc. The list for these IOC's will be created with domain experts in security (our client) and then stored in a table in our database. Tweets with IOC's found will be assigned the IOC to their respective columns in the database.

#### *2.2.2*

For clustering on similar tweets, I plan to apply machine learning to cluster tweets based on similarities. The goal for this will be to have a method to pull all the tweets from the database based on how semantically similar they are. One way to do this is by using the Jaccard similarity which will literally cluster based on the number of common words between Tweets. This is a very bad method for clustering as some tweets maybe talking about the same attack but with very few to none common words in them leading to these being classified in different clusters. Another method is pre-trained sentence encoders which is similar to word embeddings like glove word2vec but for encoding sentences as opposed to encoding words. This essentially creates a matrix with the names for rows and columns being sentences (row i and column i's label will correspond to the same sentence) that we feed in as training input and a

correlation between all the different sentences (as a probability) as values in the matrix. This has been one of the best methods for clustering sentences based on semantic similarity. The last method that we can use is the Siamese Manhattan LSTM (MaLSTM). The technical insight that makes Siamese networks better than other machine learning architectures is that the learning stage for the network is done by comparing one input with another input and calculating its similarity. This method is also known as one shot learning. Manhattan LSTM just refers to the fact that we use the Manhattan distance to compare the distance between the final hidden states for the LSTM. Looking at all the methods above, I will be going with the Siamese Manhattan LSTM as this requires much fewer training examples to train, is the current state of the art and is also computationally efficient as it shares the network parameters for the two models.

### *2.3 Tweet Relevance Machine Learning Algorithm*

My part in this is to not make the final algorithm we use for classifying Tweets, with cybersecurity incidents, as relevant or irrelevant but to test different architectures and recommend the final architecture. This is to test other architectures that might give us a better accuracy rate or lower training time. I mainly plan to compare a LSTM (Long Short Term Memory Network), GRU (Gated Recurrent Unit) and a Transformer. This part will basically involve quickly prototyping each architecture, creating a baseline for each and verifying if the final architecture we use is the right architecture. LSTM's and GRU's are recurrent architecture models with the difference between them being that GRU's use lesser parameters for learning by omitting some features of the LSTM often bringing down training time and in some cases even converging to the optimal accuracy faster than a LSTM. A Transformer is a new type of neural network architecture that focuses more on the meaning between all the words in a sentence and find important words in a sentence to every word in the sentence. While my role will not be coding a transformer, it will be vetting out the practical performance of other models to the transformer architecture.

### *2.4 Graph Database*

My part will also be creating the final database and only database that will be used in production. I will be using Neo4j as my graph database of choice due to its market leading position and ease of installation. The graph database will essentially take nouns obtained from NLTK and create nodes counting occurrences of the nouns every day over a short period of time for checking if a tweet is an emerging threat or not. Apart from that, the author will also be stored as a link with the author's name and username as metadata and relations from this node to nodes with the nouns that the author used. Relationships will have the authors name and time, date for the tweet made in their metadata along with the URL for the tweet. This can be uniquely used to see tweeting frequency of a user and classify users as bots or corporate accounts.

## 3. REFERENCES

[1] A. Sieg, "Text Similarities : Estimate the degree of similarity between two texts," *Medium*, 14-May-2019. [Online]. Available: https://medium.com/@adriensieg/text-similarities-da019229c894. [Accessed: 08-Nov-2019].

[2] H. Lamba, "One Shot Learning with Siamese Networks using Keras," *Medium*, 17-Feb-2019. [Online]. Available: https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d. [Accessed: 08-Nov-2019].

[3] M. Allard, "What is a Transformer?," *Medium*, 01-Jul-2019. [Online]. Available: https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04. [Accessed: 08-Nov-2019].

[4] N. Soni, "9 Popular Cloud-based Web Scraping Solutions," *Geekflare*, 04-Nov-2019. [Online]. Available: https://geekflare.com/web-scraping-tools/#Web-Scraper. [Accessed: 08-Nov-2019].

[5] "WebScraper," *WebScraper*. [Online]. Available: https://webscraper.io/. [Accessed: 08-Nov-2019].

[6] "ScrapeStack - Free Proxy & Web Scraping API," *scrapestack*. [Online]. Available: https://scrapestack.com/. [Accessed: 08-Nov-2019].

[7] "Documentation - Web Scraper," *Text selector | Web Scraper Documentation*. [Online]. Available: https://webscraper.io/documentation/selectors/text-selector. [Accessed: 08-Nov-2019].

[8] "Web Scraping for Developers," *ParseHub*. [Online]. Available: https://parsehub.com/web-scraping-for-developers. [Accessed: 08-Nov-2019].

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz

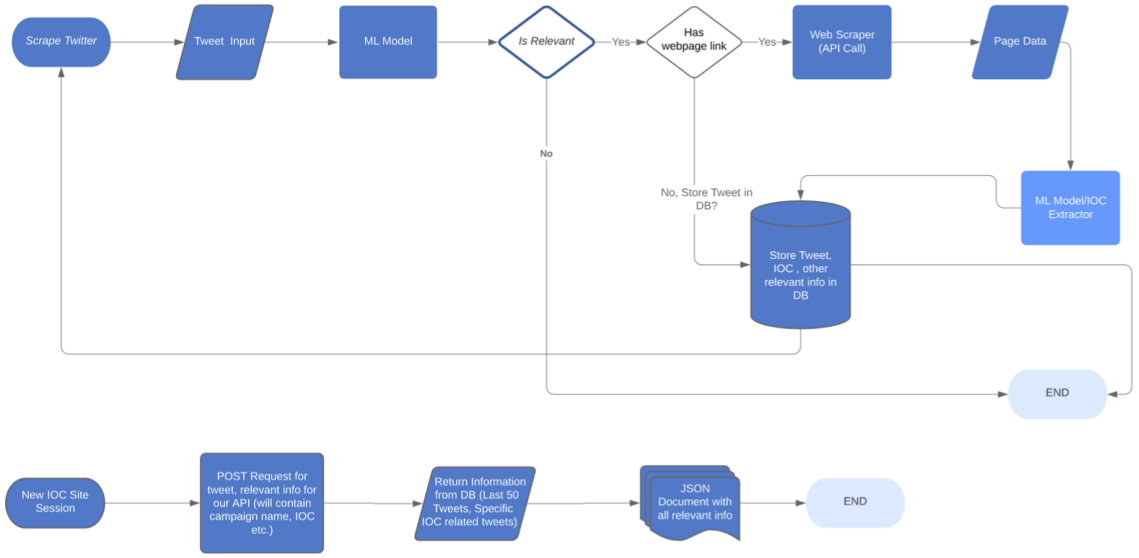
Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008

# 6 Project Documentation

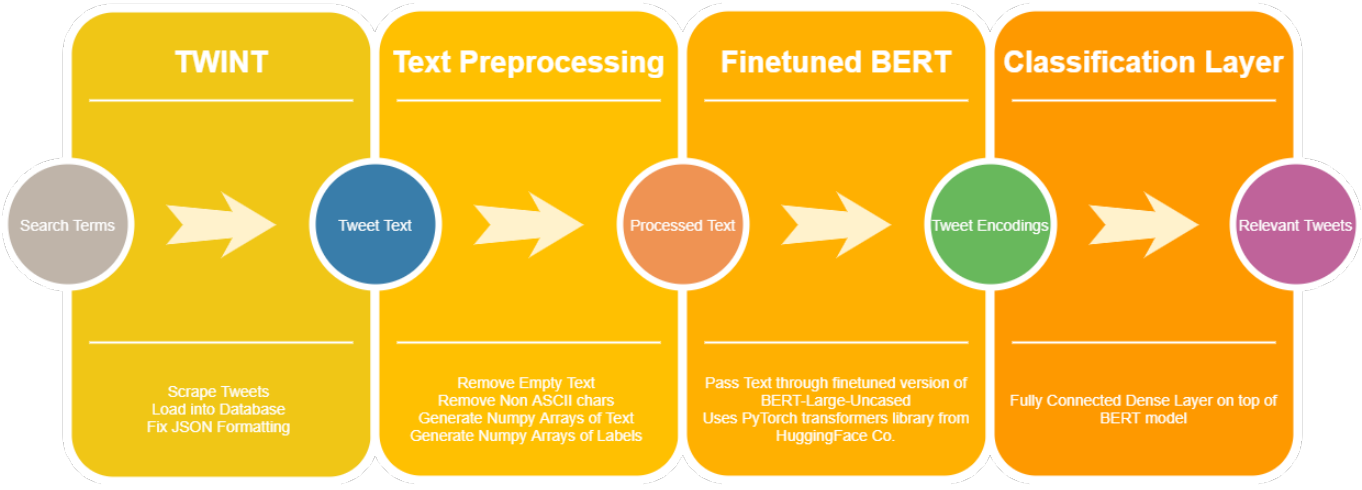
## 6.1 How it works

This diagram shows the high level interplay between the components of the project. At the simplest level, the project scrapes tweets from Twitter and passes them to the machine learning model to determine if the tweets are relevant. Relevant tweets are then passed to a noun extractor that isolates the indicators of compromise (IOC). These IOC's are then sent to the graph database to learn and store the relationships between them.

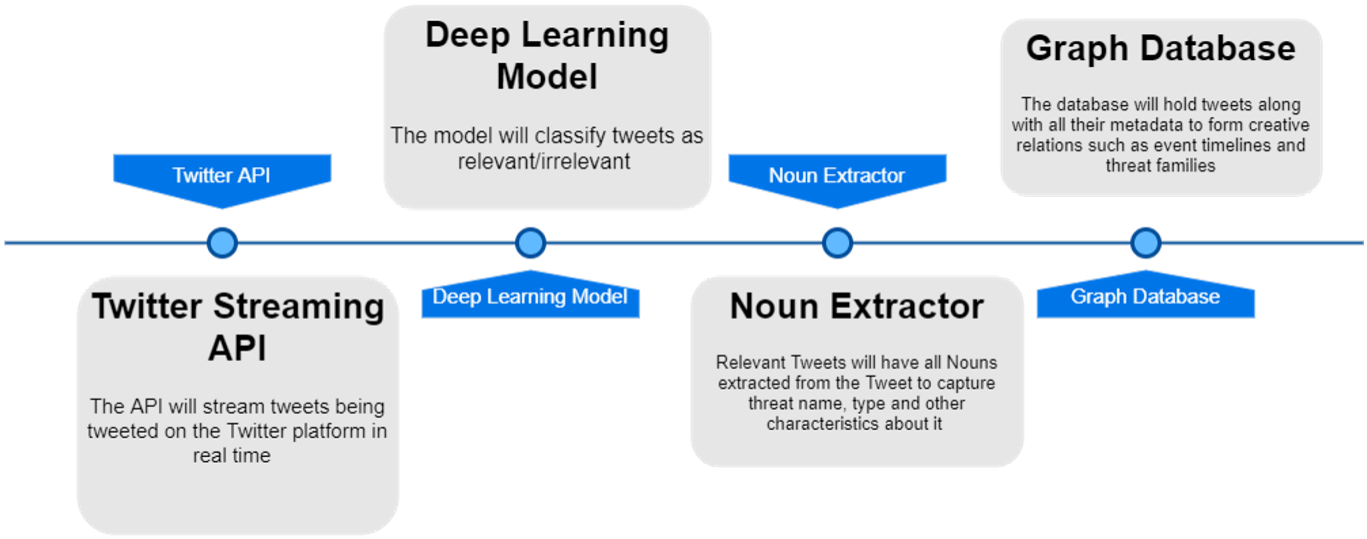
Fig. 3. High Level Component Diagram




Taking a closer look at the machine learning model, the model's pipeline performs preprocessing and then classification. The preprocessing steps ensure non-ASCII characters,URL's, and empty tweets are removed. The classification steps then pass the preprocessed text through the BERT model and the final classification layer to get a determination on a tweet's relevancy.




Once relevant tweets are identified, they are sent to a noun extractor that isolates outs the names of threat campaigns and indicators of compromise. These are then passed for storage and analysis in a graph database.


Deep Learning Model
The model will classify tweets as relevant/irrelevant
Graph Database
The database will hold tweets along with all their metadata to form creative relations such as event timelines and threat families
Twitter API
Noun Extractor
Twitter Streaming API
The API will stream tweets being tweeted on the Twitter platform in real time
Deep Learning Model
Noun Extractor
Relevant Tweets will have all Nouns extracted from the Tweet to capture threat name, type and other characteristics about it
Graph Database

## 6.2 Installation

### 1. PyTorch

Please use the selector on the PyTorch website to get the command appropriate for your system PyTorch Installation Website

### 2. Hugginface Transformer - PyTorch

Install Transformers using:

```
$ pip install transformers
```

Load Pre-trained model:

```
>>> # Load pre-trained model tokenizer (vocabulary)
>>> tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

### 3. Spacy

Install Spacy using:

```
$ python -m spacy download en
$ python -m spacy download en_core_web_sm
```

After this simply import and have fun

```
>>> import spacy
>>> nlp = spacy.load('en_core_web_sm')
```

Note: Functions for such present in final_pipeline.py. Check in-file documentation on calling them

### 4. NLTK

Install NLTK using:

```
$ pip install --user -U nltk
```

Download NLTK data using

```
>>> import nltk
>>> nltk.download()
```

Note: Functions for such present in final_pipeline.py. Check in-file documentation on calling them.

### 5. Neo4j

Install Python API using:

```
$ pip install neo4j
```

To establish connection to DB:

```
>>> from neo4j import GraphDatabase
>>> driver = GraphDatabase.driver("bolt://localhost",auth=("test_user","password"), encrypted=False)
```

Note to be functionally compliant with final_pipeline.py, create user and password "test_user" and "Password" respectively

### 6. Twint

Install Using

```
$ pip3 install twint
```

Note: Functions for such present in the final_pipeline.py. Check in-file documentation on calling them.

### 7. Twitter API

Requires getting approved for a Twitter developer account. Streaming API requires a paid version (Enterprise) of the Twitter API. We have a basic connection to the twitter API established.

Twitter documentation for authenticating/connecting to Twitter API - Most time consuming step

Here is a list of our keys that might help in looking for what keys should look like:

```
>>> oauth_consumer_key = ZYOj2R1VYsIUhaRvsrcRuHxk6
>>> oauth_nonce = kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg
>>> Consumer Secret = API Secret Key = uTwMxW2CPXfv8wZCcqOSX8Yw46eqJVMNopqh5kh9O0VXiVQNRG
>>> Access token secret = OAuth token secret: VB6fxb1akdtwRDf52sbQlPKE9tpHEPX1ZBSI3Y1t33REk
>>> Signing key = uTwMxW2CPXfv8wZCcqOSX8Yw46eqJVMNopqh5kh9O0VXiVQNRG&VB6fxb1akdtwRDf52sbQlPKE9tpHEPX1ZBSI3Y1
>>> oauth-token = 797644398670409728-I3aUqpmhD7uPscfFZFxMGfMWXPBmiaN
>>> oauth_signature = Use HMAC
```

Establishing a connection:

```
>>> import twitter
>>> api = twitter.Api(consumer_key="ZYOj2R1VYsIUhaRvsrcRuHxk6",
                      consumer_secret="uTwMxW2CPXfv8wZCcqOSX8Yw46eqJVMNopqh5kh9O0VXiVQNRG",
                      access_token_key="797644398670409728-Zwgcl9kcCFerhFNlFFGwR3emSbfpfpX",
                      access_token_secret="CzVCCqD8X9FC059X98deDiNYb24IWjhZYVeAhoU4F5v7l")
```

### 6.3 Running the Pipeline

#### Instructions on Running

1. Make sure that your security_tags.txt file and model.pt file are stored in your working directory.
2. Run the pipeline using either option 1 or option 2 discussed below.

Pipeline takes in the following parameters

- An Option
- A data source
- The model
- The graphDB username
- The graphDB password

Running the Pipeline

Option 1 - This is used for running the pipeline when you want to pass in the tweets through a json file. To run the Pipeline this way the command would be:

```
python pipeline.py {1} {json File} {model file} {graphDB username} {graphDB password}
```

Option 2 - This is used for running the pipeline when you want to use the Twint Api to collect tweets to pass through the pipeline. The command for option 2 would be:

```
python pipeline.py {2} {hashtags txt file} {model file} {graphDB username} {graphDB password}
```

### 6.4 Useful Links and API documentation

- PyTorch Documentation Link
- Hugginface Transformer Documentation Link
- Spacy Documentation Link
- NLTK Documentation Link
- Neo4j Documentation Link
- Twint Documentation Link
- Twitter API documentation Link

## 7 Recommended Technical Resources for Learning More

- Information about Social Media Intelligence related to Cybersecurity on Twitter
- Learning about what BERT

## 8 Poster

# AI BASED THREAT INTELLIGENCE

Automating the process of identifying cybersecurity campaigns and other high-profile threats on Twitter

## INTRODUCTION

The client for this project is McAfee, a company that focuses on cybersecurity. They produce products that protect computing systems against malware and other threats.

One of the largest challenges they face is ensuring that their products are up to date and capable of defending against the latest threats. To meet these requirements in a rapidly evolving cybersecurity landscape, the client continuously performs thorough research; needless to say, this constant search for relevant news is an arduous task. Our aim for this project is to make this research process less labor intensive by automating portions of it.

## BACKGROUND

The problem that all cybersecurity companies face is that they are never in a state of peace. Attackers are constantly looking for ways to exploit vulnerabilities and are always looking for backdoors to engage in criminal activities.

It is our client, McAfee's job to try and prevent these attackers from being able to cause harm on their consumers. With over 400 new threats being discovered every minute, it is important for McAfee to stay up to date with what potential threats exist, and which threats they need to focus on to make sure they can protect their consumers the best.

Part of staying up to date with threats and protecting their consumers includes having teams of researchers identifying new threats and reverse engineering them to ensure that McAfee's products are able to protect against them.

## SUMMARY OF STEPS

We first began by scraping tweets into an AWS database. We then manually labelled these tweets based on their relevance to cybersecurity threats. The resulting dataset was used to train a deep learning model that can filter tweets and provide the relevant ones. In our final pipeline, we used the twitter streaming API to feed tweets to our deep learning model. The tweets identified as relevant were then put through a noun extractor and stored in a graph database to provide visuals about emerging threats and their relations.

Oregon State University

## PROJECT DESCRIPTION

### 1. TWITTER SCRAPING AND DATASET CREATION

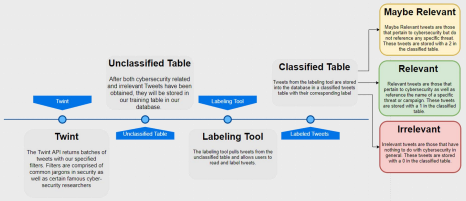



Figure 1: This figure shows the steps that were taken for gathering and labeling Tweets.

### 2. TWEET CLASSIFICATION VIA DEEP LEARNING

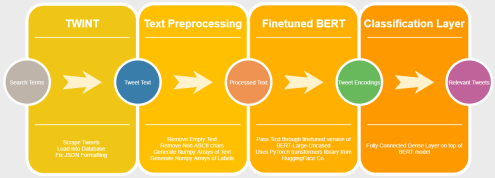



Figure 2: This figure shows what steps are taken from the time a tweet is collected to the time it runs through the classifier.

### 3.FEATURE EXTRACTION/END PRODUCT PIPELINE

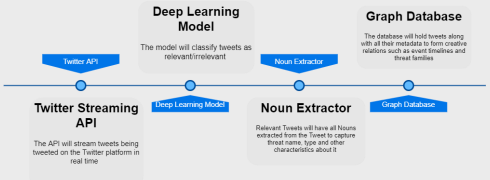



Figure 3: This figure displays the end product pipeline that will be used by McAfee LLC to optimize their threat research.

## ACKNOWLEDGEMENTS

## TEAM MEMBERS

Harshvardhan Singh – Singhhar@oregonstate.edu

Nikhil Anand – Anandn@oregonstate.edu

Rohan Varma – Varmar@oregonstate.edu

## CLIENTS

Prashanth Ramagopal –Principal Engineer at McAfee LLC

Sameer Paranjape – Principal Engineer at McAfee LLC

Simon Ezequias – Security Researcher at McAfee LLC

## RESULTS

### DATASET CREATION

Over the course of the project we classified 5002 tweets pulled in from a mixture of cybersecurity related users as well as completely random users. We ended up creating a dataset consisting of 917 relevant tweets, 552 maybe relevant tweets, and 3533 irrelevant tweets. Out of the total 5002 tweets, we took out 25% of them to be our test set. We maintained the 30-70 ratio of related to unrelated within both the training and testing datasets.

### CLASSIFICATION

Our final finetuned BERT model took the pretrained BERT-Large-Uncased model and finetuned it for 3 epochs on our training dataset. It performed very well on our test set, recording an accuracy of 91% and an AUC of 0.95 (AUC has a maximum of 1, and AUC values above 0.9 demonstrate excellent performance).

### FEATURE EXTRACTION

We created a Noun Extraction tool using Python's Natural Language Toolkit. We then put tweets identified as relevant by the deep learning model through the Noun Extraction Tool and stored the resulting characteristics (not just nouns) in a Neo4j graph database. The graph database formed intuitive relations on certain attributes such as nouns and date published to give visual groupings and timelines of events for threats.

McAfee

# 9 Conclusions and Reflections

## 9.1 Nikhil Anand

I learned a ton over the course of the project, I learned things related to Machine Learning such as what Bert is, and other tools and libraries for Natural Language Processing. I also learned how to use a Graph Database and how to create and deploy a product pipeline. In terms of non-technical learning, I learned a lot about how to separate responsibilities for a task and how to break down goals into very small modularized tasks to get work done efficiently. I learned that with project work, it may seem that everything will just get done faster, but there can also be hurdles along the way as everyone thinks differently, so we may not always agree on the approach and finding ways around these hurdles is important for keeping the team together and functioning efficiently and successfully. Project Management was something I hadn't thought about a lot throughout the process but thinking back on the project, I can see how important project management is, because taking a step back and looking at the big picture to accomplish our goals is really what helped us get through some challenges, so had we done that earlier we may have been able to avoid some difficulties we had throughout the way. Like I said before I believe most people enjoy team projects because it helps distribute the stress of completing the project, but I have learned that to work successfully as a team there has to be a lot of time spent on keeping organized because when multiple people are all working on the same goal, things tend to get messy and out of order so making sure you are on top of things is critical. If I could do it all over again, I would take the research part of Fall term more seriously, because a lot of the things we thought we would do one way, ended up getting implemented differently which made the other two terms much more stressful.

## 9.2 Rohan Varma

As someone hoping to pursue a career in machine learning, being the machine learning lead for this project was very enlightening. The project was my first real experience with trying to create a model that performed well on custom tasks that hadn't already been researched, and this exposed me to several challenges I did not have to deal with earlier. Firstly, I realized that in the real world, the largest determining factor of how good of a model you can produce is the quality of your dataset - and acquiring a good quality dataset is often very difficult due to a variety of factors ranging from manpower available to consistency in dataset curation. Additionally, this was my first time experiencing the requirement planning portion of a project, and I came to see how important it is that clients and customers be on the same page; midway through the first term, the scope of the project was greatly expanded, and it made us completely reevaluate our timelines and approach. If I were to do it over again, I think I definitely would've placed more emphasis on outlining the exact requirements. I would also have placed more emphasis on clearly outlining the deliverables for each team member, as I came to see that without clearly defined roles it is often unclear what each team member should be working on. That said, ultimately this project showed me that real work in the real world is very dynamic and you need to roll with the punches and do whatever it takes to satisfy the client.

## 9.3 Harshvardhan Singh

What technical information did you learn? There were a lot of exciting technical components that I learned to work with. Some of the selected ones are Data Curation, Feature Engineering, Semantic Word Extraction and Graph Analytics. Apart from this I learnt on how to maintain and create valuable documentation for sustaining projects and modularizing code. Project work is a group thing and for a group to be successful, every member should be on the same page in terms

of understanding what the final goal of the project is and to successfully implement parts of a project for it to work together. Being the team lead for the project, I understand the importance in having everyone have concrete parts of a system to work on and having deadlines for people to follow for project success. Another important thing would be having a document to actively track what components of a system are being developed and fully complete architectural documents before implementing it. Resolving conflicts within a group is also essential to the success of a project. Apart from all this, I learned on working with a group and communicating with our client to formulate requirements and work with a team using the agile process. Some of the things that I would've done if I can do this again is holding people to their commitments. Having everyone have clear parts is essential for success in a group. Holding people accountable to their commitments is also something that is necessary for all group members to succeed in a project and even for the overall success of the group. On the technical part, organizing my time would have better helped me accommodate to changing schedules. Also, having a very simple paragraph at the top of every code file can go a long way in making sure that everyone's code is readable and re-implementable for a long time. What non-technical information did you learn? What have you learned about project work? What have you learned about project management? What have you learned about working in teams? If you could do it all over, what would you do differently?

## 10 APPENDIX

### 10.1 Appendix 1

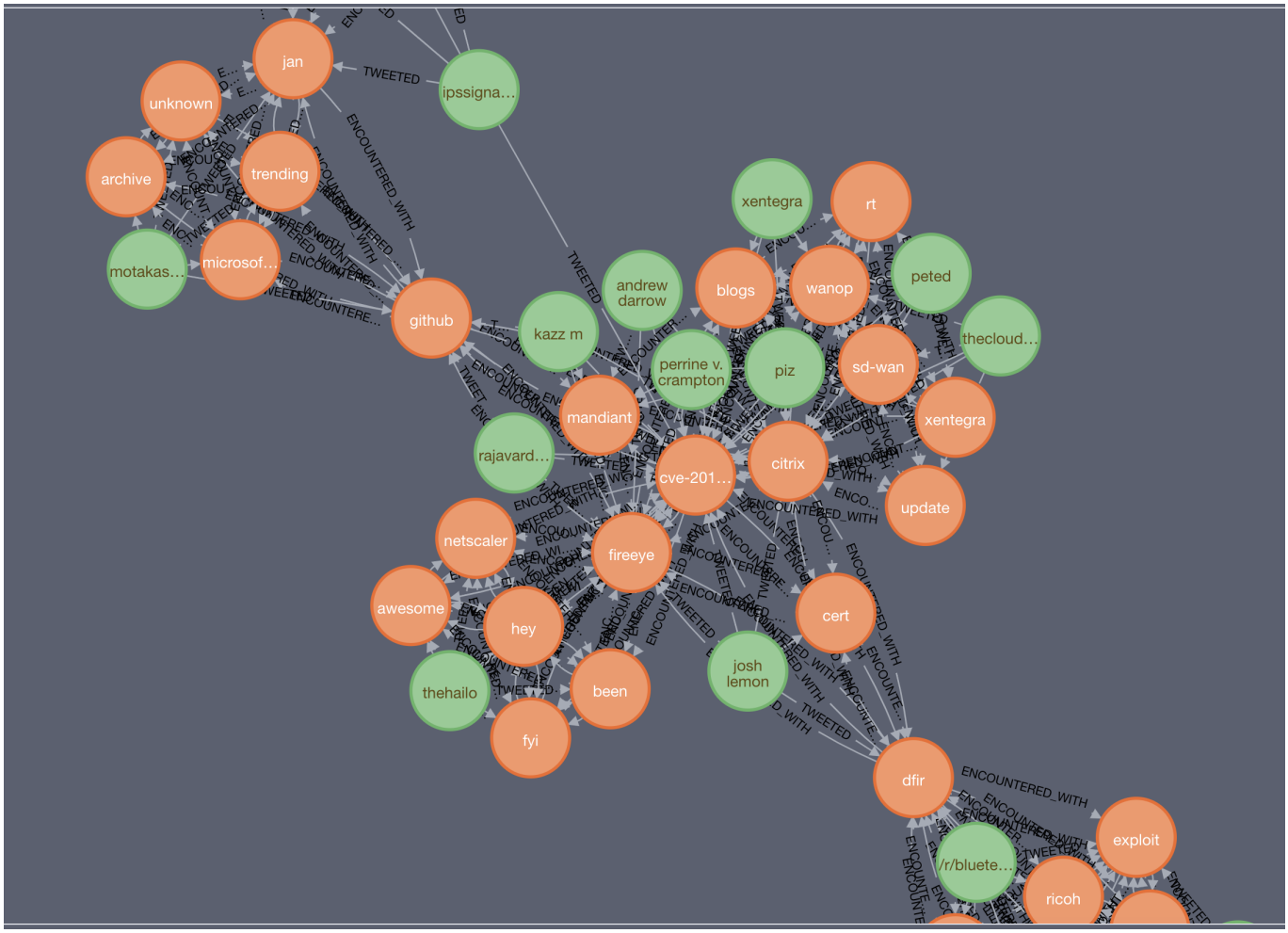



Fig 4. A picture displaying nouns from relevant tweets from Twitter from various authors being modeled as a graph

### 10.2 Appendix 2

Code for Creating Nodes and Relations in Graph

```
def addToGraph(driver, nouns, date, tweet_id, link, username, name, time):
    nouns  = list(set(nouns))
    with driver.session() as session:
        tx = session.begin_transaction()
        count = 0
        for noun in nouns:
            result = tx.run('''Match (a:Tweet {type: $word}) return a''', word=noun.lower())
            result = result.records()
            flag = True
            for record in result:
                flag = False
                find_index = -1
                for i in range ( len( record['a']['last10DaysDate'] )):
                    if record['a']['last10DaysDate'][i] == date:
                        find_index = i
                        break
                if find_index != -1:
                    tx.run('''MATCH (a:Tweet {type:$word}) SET a.num = a.num+1, a.last10DaysCount
=
                    a.last10DaysCount[ ..$index ] + (a.last10DaysCount[ $index ] + 1) + a.last10D
aysCount[ $index+1.. ], a.link_id = a.link_id + [ $link_id ] ''', word=noun.lower(), index = find
_index, link_id = link + " " + tweet_id)
                else:
                    tx.run('''MATCH (a:Tweet {type:$word}) SET a.num = a.num+1, a.last10DaysDate
= a.last10DaysDate + [$date], a.last10DaysCount = a.last10DaysCount + [1], a.link_id = a.link_id
+ [ $link_id ] ''', word=noun.lower(), date=date, link_id = link + " " + tweet_id)
            if flag:
                tx.run('''CREATE (a:Tweet {type:$word, num:1, last10DaysDate:[$date], last10DaysC
ount:[1], link_id:[ $link_id ]})''', word=noun.lower(), date=date, link_id = link + " " + tweet_i
d)
            tx.run('''MERGE (a:Author {username:$username, name:$name})''', username=username.low
er(), name = name.lower())

        for i in range(len(nouns)):
            tx.run('''
                    Match (a:Author { username: $username }), (b:Tweet {type: $noun})
                    MERGE (a)-[: TWEETED {date: $date, time: $time}]->(b)
                    ''', noun=nouns[i].lower(), date=date, time=time, username=username.lower)

            for j in range(i+1, len(nouns)):
                if nouns[i] == nouns[j]:
                    continue
                tx.run('''
                        Match (a:Tweet { type: $noun_1 }), (b:Tweet {type: $noun_2})
                        MERGE (a)-[: ENCOUNTERED_WITH {type: $noun_2}]->(b)
                        MERGE (a)<-[: ENCOUNTERED_WITH {type: $noun_1}]-(b)
                        ''', noun_1=nouns[i].lower(), noun_2=nouns[j].lower())

        tx.commit()
```

Code for checking whether a noun represents an emerging threat

```
def check_relevance (driver, noun):
    with driver.session() as session:
        tx = session.begin_transaction()
        result = tx.run('''Match (a:Tweet {type: $word}) return a''', word=noun.lower())
        result = result.records()
        ans = []
        for record in result:
#             print(sorted(zip(record['a']['last10DaysDate'], record['a']['last10DaysCount'])))
            date_and_count = sorted(zip(record['a']['last10DaysDate'], record['a']['last10DaysCou
nt']))
            if (date_and_count[-1][1] > 3):
                link_id =  (*map( lambda x: x.split(" "), record['a']['link_id'] ), ) # Contains
all urls encountered for that noun
                for i in range(len(link_id)):
                    print(link_id[i][0].split('/status')[0] + '/status/' + link_id[i][1], noun, d
ate_and_count[-1][1])
#                 print(link_id)
                print("\n")

link_id = check_relevance(driver, 'mp4')
```

Code for loading the BERT model

```
# Load BertForSequenceClassification, the pretrained BERT model with a single
# linear classification layer on top.
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased", # Use the 12-layer BERT model, with an uncased vocab.
    num_labels = 2, # The number of output labels--2 for binary classification.
                    # You can increase this for multi-class tasks.
    output_attentions = False, # Whether the model returns attentions weights.
    output_hidden_states = False, # Whether the model returns all hidden-states.
)
model.load_state_dict(torch.load(modelFile,map_location=device))
model.eval()
model.to(device)
```

### 10.3 Appendix 3

The feedback we got from our code review with our client was the following:

- Lack of Unit Tests
- Code for pipeline not modularized into seperate files
- Github home directory was quite cluttered with files that were no longer in use
- README didn't discuss dependencies and how to run the pipeline

We addressed the first critism by adding unit tests that would be used to ensure that the different modules within the pipeline were working correctly. We then modularized the pipeline into the following files: Twitter.py which is used for scraping tweets, pipeline.py which is used to run the pipeline, nlp.py which has all the functions that are used to extract nouns from relevant tweets, MLmodel.py which has all the functions that are used to classify tweets as relevant or irrelevant, graphdb.py which has all the functions related to adding nouns to the graph database and

classifying emerging threats, dataprocessing.py which has functions used to clean up text before it is run through the pipeline. There is also a unittest.py which has our unit tests for the pipeline. We then cleaned up our Github directory by adding in relevant folders to organize the files better, and added in step by step instructions in the README for all dependencies and functionality involving the pipeline.